UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie mécanique

EXPLORATION AUTONOME ET EFFICIENTE DE CHANTIERS MINIERS SOUTERRAINS INCONNUS AVEC UN DRONE FILAIRE

ENABLING EFFICIENT AUTONOMOUS EXPLORATION OF UNKNOWN UNDERGROUND MINING STOPES USING A TETHERED AERIAL ROBOT

Thèse de doctorat
Specialité: génie mécanique

Louis PETIT

Sherbrooke (Québec) Canada

Mars 2023

JURY MEMBERS

Alexis LUSSIER DESBIENS

Supervisor

David RANCOURT

Examiner

François GRONDIN

Examiner

Philippe HAMELIN

Examiner

RÉSUMÉ

La cartographie des chantiers miniers souterrains est souvent réalisée à l'aide d'un capteur situé au bout d'une perche que l'opérateur introduit dans le chantier, depuis une zone sécurisée. Le capteur émet des faisceaux laser qui fournissent la distance à un mur détecté, créant ainsi une carte en 3D. Ceci produit des zones d'ombres et une faible densité de points sur les parois éloignées. Pour relever ces défis, une équipe de recherche de l'Université de Sherbrooke conçoit un drone filaire équipé d'un LiDAR rotatif pour cette mission, bénéficiant ainsi de plusieurs points de vue. La transmission filaire permet un temps de vol illimité, un partage de calcul et une communication en temps réel. Pour une compatibilité avec le mouvement du drone lors des coincements du fil, la longueur excédante est intégrée dans une bobine embarquée, qui contribue à la charge utile du drone. Lors d'un pilotage manuel, le facteur humain entraîne des problèmes de perception et compréhension d'un environnement 3D virtuel, et d'exécution d'une mission optimale.

Cette thèse se concentre sur la navigation autonome sous deux aspects : la planification de trajectoire et l'exploration. Le système doit calculer une trajectoire qui cartographie l'environnement complet, en minimisant le temps de mission et en respectant la longueur maximale de fil embarquée. La planification de trajectoire à l'aide d'un *Rapidly-exploring Random Tree* (RRT) trouve rapidement un chemin réalisable, mais l'optimisation est coûteuse en calcul et la performance est variable et imprévisible. L'exploration par la méthode des frontières est représentative de l'espace à explorer et le chemin peut être optimisé en résolvant un *Traveling Salesman Problem* (TSP), mais les techniques existantes pour un drone filaire ne considèrent que le cas 2D et n'optimisent pas le chemin global.

Pour relever ces défis, cette thèse présente deux nouveaux algorithmes. Le premier, *RRT-Rope*, produit un chemin égal ou plus court que les algorithmes existants en un temps de calcul jusqu'à 70% plus court que le deuxième meilleur algorithme dans un environnement représentatif. Une version modifiée de *RRT-connect* calcule un chemin réalisable, raccourci avec une technique déterministe qui tire profit des noeuds intermédiaires préalablement ajoutés. Le deuxième algorithme, *TAPE*, est la première méthode d'exploration de cavités en 3D qui minimise le temps de mission et la longueur du fil déroulé. En moyenne, le trajet global est 4% plus long que la méthode qui résout le TSP, mais le fil reste sous la longueur autorisée dans 100% des cas simulés, contre 53% avec la méthode initiale. L'approche utilise une architecture hiérarchique à 2 niveaux : la planification globale résout un TSP après extraction des frontières, et la planification locale minimise le coût du chemin et la longueur de fil via une fonction de décision.

L'intégration de ces deux outils dans le *NetherDrone* produit un système intelligent pour l'exploration autonome, doté de fonctionnalités semi-autonomes pour une interaction avec l'opérateur. Les travaux réalisés ouvrent la porte à de nouvelles approches de navigation dans le domaine des missions d'inspection, de cartographie et de recherche et sauvetage.

Mots-clés : robots autonomes, cartographie, drones, planification de trajectoire, exploration, systèmes intelligents, rapidly-exploring random trees, traveling salesman problem.

ABSTRACT

Underground mining stopes are often mapped using a sensor located at the end of a pole that the operator introduces into the stope from a secure area. The sensor emits laser beams that provide the distance to a detected wall, thus creating a 3D map. This produces shadow zones and a low point density on the distant walls. To address these challenges, a research team from the Université de Sherbrooke is designing a tethered drone equipped with a rotating LiDAR for this mission, thus benefiting from several points of view. The wired transmission allows for unlimited flight time, shared computing, and real-time communication. For compatibility with the movement of the drone after tether entanglements, the excess length is integrated into an onboard spool, contributing to the drone payload. During manual piloting, the human factor causes problems in the perception and comprehension of a virtual 3D environment, as well as the execution of an optimal mission.

This thesis focuses on autonomous navigation in two aspects: path planning and exploration. The system must compute a trajectory that maps the entire environment, minimizing the mission time and respecting the maximum onboard tether length. Path planning using a *Rapidly-exploring Random Tree* (RRT) quickly finds a feasible path, but the optimization is computationally expensive and the performance is variable and unpredictable. Exploration by the frontier method is representative of the space to be explored and the path can be optimized by solving a *Traveling Salesman Problem* (TSP) but existing techniques for a tethered drone only consider the 2D case and do not optimize the global path.

To meet these challenges, this thesis presents two new algorithms. The first one, *RRT-Rope*, produces an equal or shorter path than existing algorithms in a significantly shorter computation time, up to 70% faster than the next best algorithm in a representative environment. A modified version of *RRT-connect* computes a feasible path, shortened with a deterministic technique that takes advantage of previously added intermediate nodes. The second algorithm, *TAPE*, is the first 3D cavity exploration method that focuses on minimizing mission time and unwound tether length. On average, the overall path is 4% longer than the method that solves the TSP, but the tether remains under the allowed length in 100% of the simulated cases, compared to 53% with the initial method. The approach uses a 2-level hierarchical architecture: global planning solves a TSP after frontier extraction, and local planning minimizes the path cost and tether length via a decision function.

The integration of these two tools in the *NetherDrone* produces an intelligent system for autonomous exploration, with semi-autonomous features for operator interaction. This work opens the door to new navigation approaches in the field of inspection, mapping, and *Search and Rescue* missions.

Keywords: autonomous robots, mapping, unmanned aerial vehicles, path planning, exploration, intelligent systems, rapidly-exploring random trees, traveling salesman problem.

"[This] is about connecting the dots."
Steve Jobs

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest thanks and gratitude to my supervisor, Pr. Alexis Lussier Desbiens. His continuous support was a key element in the success of this project. His willingness to understand all aspects of robotic systems, his insight and his enthusiasm for scientific research led to many interesting discussions and ideas. The freedom he gave me allowed me to explore the problem and come up with the ideas that were later published in our papers. He also helped me a lot in my personal and professional development, whether it was through engaged discussions, reflections on my future as a researcher, or his support for various formalities when necessary.

I would like to thank the members of my jury, namely Pr. David Rancourt, Pr. François Grondin and Philippe Hamelin, who accepted to participate in the revision of this thesis. I would also like to thank Pr. David Rancourt for being in my supervisory committee. His good engineering sense allowed me to take a step back from my research when necessary, and it allowed me to better quantify the benefits of my algorithms.

I am grateful to Pr. François Grondin and Pr. Manuel Lafond for their time and precious help when I needed their expertise for the resolution of the Traveling Salesman Problem. Even though some of the ideas I have pursued are not part of this thesis, I ended up being passionate about this problem, and transmitting this passion in various presentations.

I would like to thank all the members of the Createk research group and my colleagues who have helped me in various ways in life and in study, and more particularly Dr. Mathieu Labbé, John Bass and Julien Rachiele-Tremblay.

Also, I would like to acknowledge the financial support from the Interdisciplinary Institute for Technological Innovation (3IT) and the Université de Sherbrooke.

Finally, I am very grateful to my loved ones who have been present in my personal life all along the way, and especially my parents. Their support and unconditional love has allowed me to move forward and carry out this project despite the obstacles in my path. They are a major element that has shaped the person I am happy to have become.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronym	Definition
2D	Two-Dimensional
3D	Three-Dimensional
AGP	Art Gallery Problem
APF	Artificial Potential Field
CMS	Cavity Monitoring System
CPU	Central Processing Unit
BFS	Breadth-First Search
BnB	Branch-and-Bound
CLK	Chained Lin-Kernighan
CTP	Canadian Traveler Problem
DFS	Depth-First Search
GBP	Graph-Based Planner
GPS	Global Positioning System
GTSP	Generalized Traveling Salesman Problem
HK	Held-Karp
LiDAR	Light Detection And Ranging
LK	Lin-Kernighan
LP	Linear Programming
MST	Minimum Spanning Tree
NBVP	Next-Best-View Planner
NN	Nearest Neighbor
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
RTAB-Map	Real-Time Appearance-Based Mapping
RTH	Return To Home
SA	Simulated Annealing
SAR	Search And Rescue
SLAM	Simultaneous Localization And Mapping
TAPE	Tether-Aware Path planning for Exploration
TARE	Technologies for Autonomous Robot Exploration
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
WRP	Watchman Route Problem

CHAPTER 1

INTRODUCTION

With the technological advances of recent years in the areas of mechanics, electronics, automation control and computer science, drones have become more and more present in our society. Today, they are mainly used for recreational or filmographic purposes. If well designed, industries can leverage this tool for applications in express delivery [Dorling et al., 2017], Search And Rescue (SAR) operations [Balta et al., 2016], agriculture services [Tripicchio et al., 2015], structure inspection [Bircher et al., 2015], wildfire monitoring [Saffre et al., 2022], surveillance [Grocholsky et al., 2006] and many others. In the context of drone mapping, the Université de Sherbrooke has been mandated by a Québec survey company to design a drone for mapping underground mining stopes. The generated 3D map allows mining companies to calculate the volume of the stope. This chapter introduces the configuration of a stope and why calculating its volume is important. The next section justifies the use of a tethered drone for stopes mapping. Next, the navigation problem and related research question are presented, along with examples of missions that justify the need for autonomous navigation. Finally, the original contributions of this thesis are presented.

1.1 Stope volume calculation

This work will focus on mapping operations of underground mining stopes. A stope is an excavated space that contains the mined ore. It is created by drilling holes in an access tunnel to place explosives and then blast them. A representation is provided in Fig. 1.1.

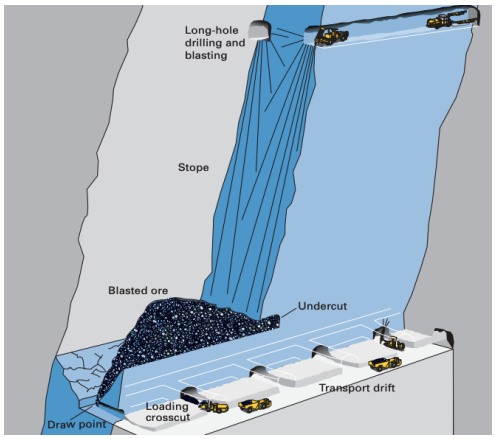


Figure 1.1 Underground mining stope layout [Atlas copco, 2022].

If the blast is oversized, the mine may be weakened or the tunnel entrance may be blocked, and human operators may be exposed to danger. The mining industry accounts for more injuries and fatalities than any other industry, according to [TheWorldCounts, 2020]. More than 15,000 people are killed each year worldwide, for reported cases only. On the other hand, an undersized blast will lead to poor extraction and a new blast will be required. In both cases, the financial losses reach several million dollars [Yuvka et al., 2020]. The calculation of stopes volume is therefore an important factor for the mining industry to optimize their excavation, extension and storage operations.

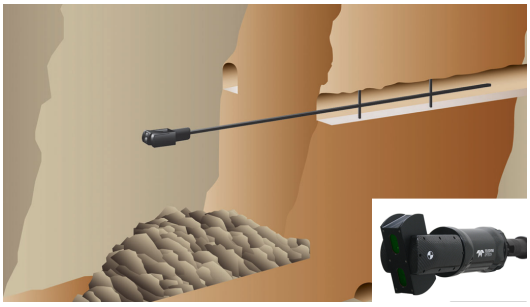


Figure 1.2 Cavity Monitoring System (CMS) [Teledyne optech, 2019].

The most common tool used by survey companies for this calculation is a Cavity Monitoring System (CMS) as displayed in Fig. 1.2. This system, placed on a 15 meters pole, allows to measure the distance to the walls and to reconstruct the 3D map of the stope by combining the data.

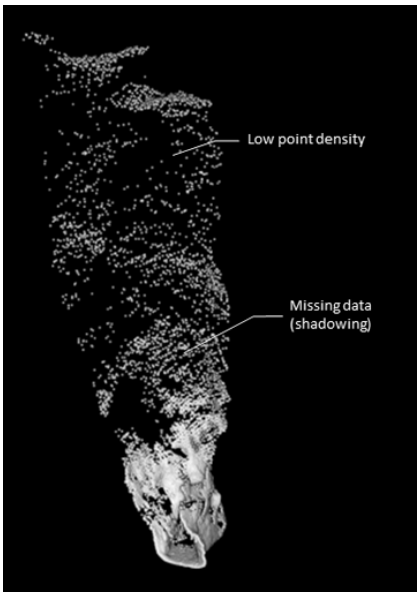


Figure 1.3 CMS scan of a mining stope [Emesent, 2019].

The CMS offers sufficient accuracy near the sensor, but the reconstruction of walls located at more than 30 meters from the sensor is unreliable, while the stopes can reach up to 300 meters deep and 100 meters wide. In addition, some areas are obstructed by rocky outgrowths on the walls. This leads to a low point density in some areas of the scan. The point cloud in Fig. 1.3 shows an example of a scan performed in a stope with this sensor. Also, the scan operation is dangerous for the operator who has to be in an area where there is a risk of falling rocks or ground collapse. Finally, the operation is time-consuming since it is necessary to place oneself at several locations to take several views and the process takes an average of 2 hours, according to [Emesent, 2019].

1.2 Tethered mapping drone

Our research project, with a team under the supervision of Professor Alexis Lussier Desbiens, aims to provide an autonomous drone capable of performing this mapping operation using a rotating *Light Detection And Ranging* (LiDAR) sensor. The rotating LiDAR provides a 360° spherical view after the assembly of successive scans. Since underground mines are GPS-denied, the LiDAR is used for localization, as well as mapping. This product allows to obtain a point density that is independent of the starting point of the drone since it can circulate to visit the shadow zones. Drone mapping therefore offers better coverage and point density. The operator of the drone can be located far from the stope, to ensure his safety. The drone mapping mission is expected to be faster than a CMS scan, with a flight time of 5 to 30 minutes. Moreover, the operator benefits from the possibility to observe the walls in detail with a camera.

Large environments such as stopes pose challenges for autonomy (i.e., limited flight time), communication with the pilot (e.g., for live inspection of the stope walls with a camera), and computation time (e.g., for path planning). These challenges can be solved with a tether intended for power and data transmission. A tethered drone benefits from an unlimited flight time, a possibility of computation parallelization and a real-time communication with the pilot. The tether must be stored in an onboard spool to allow the drone to continue navigating if the tether gets stuck in an obstacle. Since the onboard tether contributes directly to the drone's payload, keeping this mass at a reasonable value is important. The management of the tether will therefore result in constraints for the algorithms developed later in this thesis. This concept of a tethered drone has led to the design of the *NetherDrone* developed by our team, and illustrated in Fig. 1.4.

(a) (b)

Figure 1.4 NetherDrone: (a) front view, (b) rear view.

1.3 Navigation problem

This work focuses on the navigation problem of the drone. Some factors that make it difficult for a human operator to control the system are detailed in Section 1.4. The main factors are the difficulty to perceive and interpret a 3D environment, as well as to pilot in an optimal way to limit the mission time while respecting the constraints of tether unwinding and map completeness. Regarding autonomous navigation,

> "Navigation can be defined as the combination of the three fundamental competences :
> – Self-localization,
> – **Path planning**,
> – Map-building and **map interpretation**." [Nehmzow, 2003]

Seen from another perspective for the problem presented here, navigation answers the following 3 questions: *"Where am I?"* with localization and mapping (i.e., map-building), *"Where am I going?"* with exploration, and *"How do I get there?"* with path planning.

Regarding the first question, a *Simultaneous Localization And Mapping* (SLAM) approach is used to locate the Unmanned Aerial Vehicle (UAV), in addition to reconstructing the map. The Real-Time Appearance-Based Mapping (RTAB-Map) algorithm [Labbé and Michaud, 2018] was developed by the Introlab group for this purpose and is employed here. The map built by RTAB-map is used by the navigation system to interpret the environment, e.g., to locate obstacles and unexplored areas. This thesis focuses on the development of path planning and exploration algorithms for the navigation system. This system sends the desired trajectory to the UAV autopilot, described in [Dozois, 2022]. The proper functioning of the navigation algorithms is based on the reliability of the map built with RTAB-Map, whose error is assumed to be less than 30 cm at all times.

To optimize the autonomous operations of the NetherDrone, the two main requirements are the minimization of the mission time and the completeness of the map. These factors are crucial in responding to the second question, *"Where am I going?"*, and the third question *"How do I get there?"*. The resulting research question is:

> "How to enable 3D autonomous exploration of unknown underground mining stopes using a tethered drone to provide a complete map while keeping the unwinding of the tether under the maximum allowed length and minimizing mission time?"

Basically, the key elements of a successful exploration system are: a) an optimization of the global path distance, in order not to lengthen the mission time, b) a local optimization of the path to avoid tether entanglements, in order not to unroll tether unnecessarily and risk not being able to finish the mission due to lack of tether length. The next paragraphs elaborate on these specific objectives that result from the 2 primary requirements (minimization of the mission time and the completeness of the map).

For a drone that moves at near-constant speed, the mission time depends directly on the distance to be covered. Consequently, the trajectory must be as short as possible. However, if the recalculation time is too long compared to the speed of movement and map acquisition, the drone must hover for a while before continuing the mission. The mission time is therefore dependent on a second factor, which is the computation time. Appendix A provides an estimate of the mission time lost if the calculation is not performed in real time. For a mission that should last 10 minutes by constantly moving, the computation time using state-of-the-art path planning techniques leads to a mission time of 30 minutes. A minimal requirement derived from this estimate is that the calculation must be done in real time, i.e., one instance must be achieved in less time than the map update. The NetherDrone updates its map with each full rotation of the LiDAR, every 4 seconds. The number of path requests required per recalculation leads to a path request time that should be around 0.03 seconds. The main challenge of path planning in stopes is the size of the environment which greatly increases the computation time. Basically, a compromise must be made between the calculation time and the length of the path to achieve this initial requirement that will affect the duration of the mission.

The completeness of the map depends on the proper identification of the areas to be explored. Several state-of-the-art exploration methods exist to address this challenge. Nevertheless, the addition of a tether on the drone results in a requirement that involves respecting a maximum length of tether on board. Although the drone is physically compatible with tether entanglements in obstacles, thanks to its onboard spool, too many

entanglements can jeopardize the mission. As a result, the management of the tether affects the possibility of completing the mission. Navigation becomes less trivial and the existing state-of-the-art algorithms do not necessarily work anymore. For a good tether management, the tether configuration must be identified, and the paths to be taken must be evaluated based on a prediction of the tether that will be unwound.

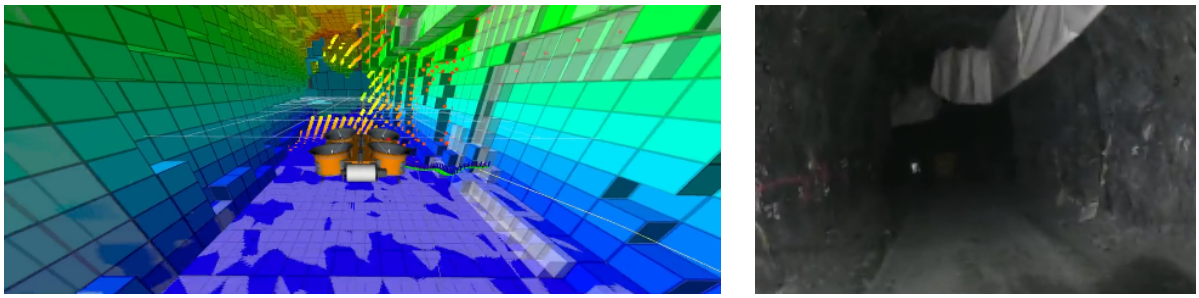
1.4 Potential applications

The approach presented in this work can be broken down into 4 distinct mission objectives that increasingly relieve the pilot of his mental load, leading to autonomous exploration:

1. Return To Home (RTH),
2. exploration assistance,
3. semi-autonomous navigation,
4. autonomous exploration.

These applications are based on real-life examples of missions where human teleoperation is complicated or not optimal. They are detailed in the following pages.

Initial mission Prior to the integration of the autonomous navigation work presented in this thesis, the NetherDrone was teleoperated by a human pilot. Since stopes are unsafe places for humans, the drone navigates out of sight. The pilot therefore has a view of a simulated 3D environment and an onboard camera, as shown in Fig. 1.5. He sends commands to the drone via a remote control. The goal of an exploration mission is to make the drone move forward in the stope, initially unknown, by discovering the walls as the mission progresses. The operation ends when the drone reaches the end of the stope and all the walls have been mapped. At the end of the mission, the pilot must bring the drone back to the take-off point.



(a) (b)

Figure 1.5 Relayed view during teleoperation: (a) virtual, (b) onboard camera.

1.4.1 Return To Home

Following a manual exploration, performed by teleoperation, the pilot may have difficulty getting the drone back to the takeoff point since the drone is out of sight during the entire mission. Fig. 1.6 shows that the tunnel entrance can be very hard to distinguish without visual aids. Also, the human pilot is not the most efficient at finding the optimal path. Finally, as much as the teleoperated exploration can be interesting to interpret the geological data, the return operation requires piloting without collecting information for almost 50% of the time. The automation of this operation is therefore beneficial. To address the question *"How to get back?"*, the first objective (see Fig. 1.7) is to implement a Return To Home (RTH) during which the drone plans a collision-free path.

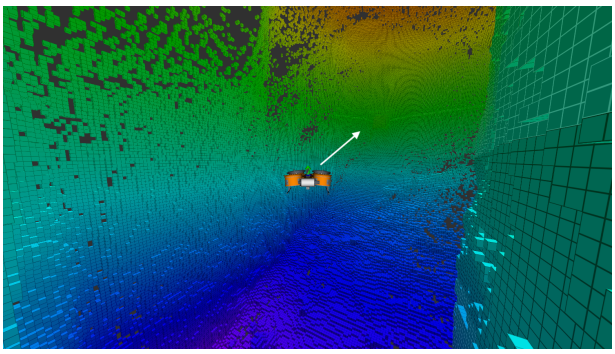
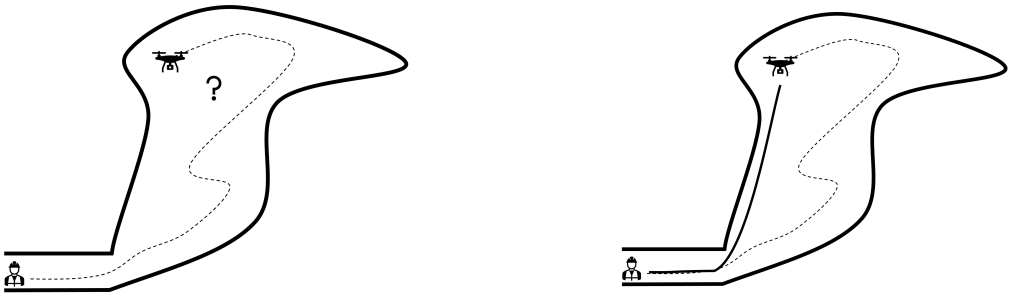


Figure 1.6 Third-person view of the drone inside the stope. The tunnel entrance is almost indistinguishable for the pilot and is located at the end of the arrow (added in post-processing).

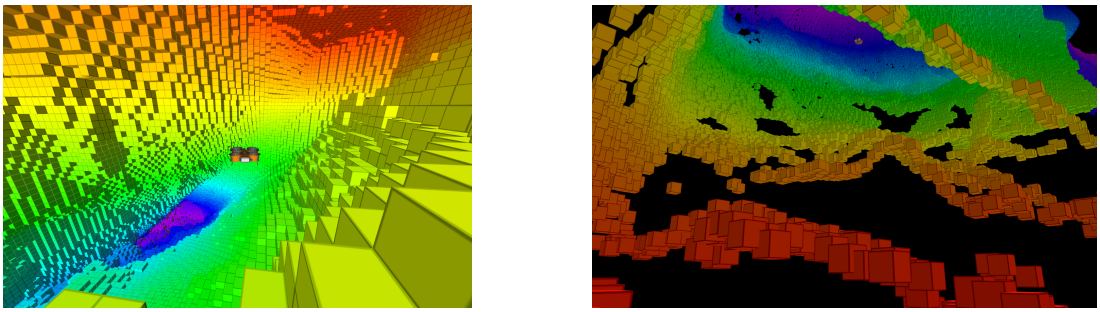


(a) (b)

Figure 1.7 Objective 1 - Return To Home (RTH). After a manual exploration flight represented by the dashed line in (a), the operator wants to find the shortest path to the home point, represented by the solid line in (b).

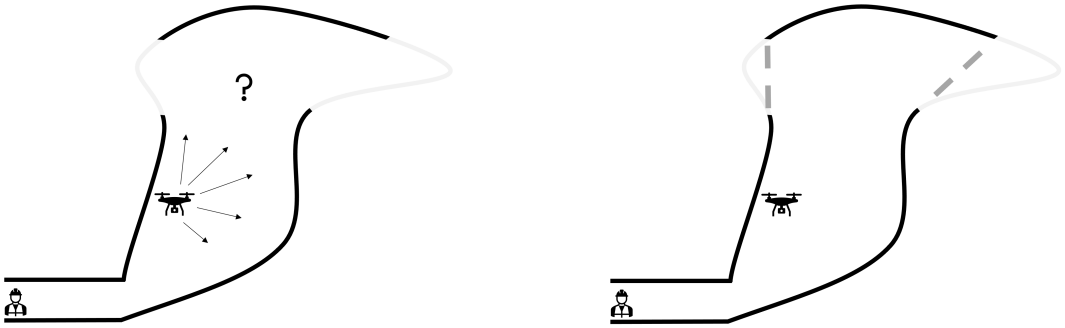
1.4.2 Exploration assistance

During exploration, it was observed during some field tests that the map often contains holes because the pilot perspective makes it difficult to distinguish unexplored spaces, as represented in Fig. 1.8. To address the question *"Where to go to discover an unexplored space?"*, the second objective (see Fig. 1.9) is to implement an exploration assistance by displaying relevant areas to the pilot.



(a) (b)

Figure 1.8 Visualization of the holes in the map: (a) the stope seen from the pilot perspective, (b) the same scan at the same time from another perspective.



(a) (b)

Figure 1.9 Objective 2 - Exploration assistance. During the manual flight in (a), the pilot wonders where to navigate, not knowing the unmapped areas, identified by the dashed line in (b).

1.4.3 Semi-autonomous navigation

Once the exploration areas have been identified, the drone must go there in 2 possible scenarios: a) the area is out of range of the LiDAR, b) rocky outgrowths obstruct part of the area to be explored. In both cases, the area to be explored is potentially a gateway to another unmapped part of the environment. It is not always easy for a human to find the shortest path, while ensuring good management of the tether. Fig. 1.10 illustrates the resulting tether configuration after a naive exploration performed by looking at the third person view only. The figure shows that the tether can easily become tangled if the pilot does not take care of it. To address the question *"How to go there?"*, the third objective (see Fig. 1.11) is to plan a path to go autonomously to the identified area.

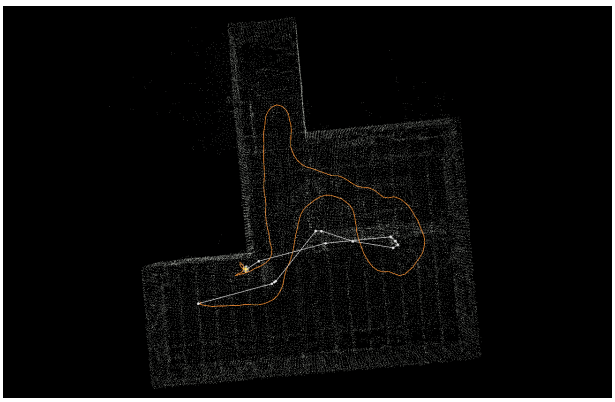
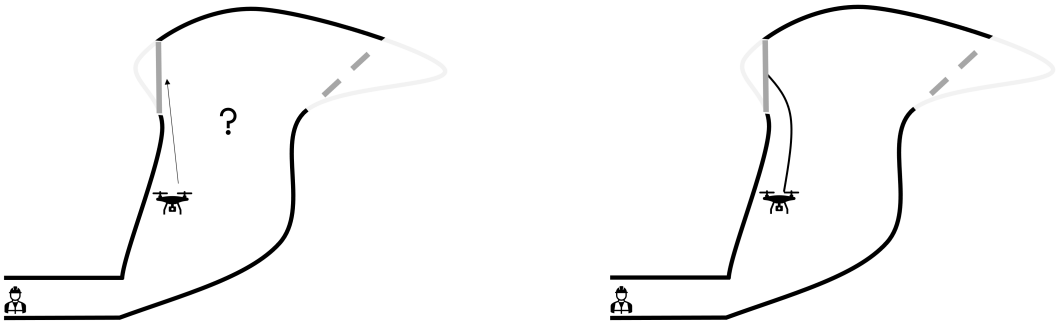


Figure 1.10 Tether configuration after a naive exploration. The tether is in gray and the trajectory in brown. The tether is entangled in the obstacles located in the middle of the map.

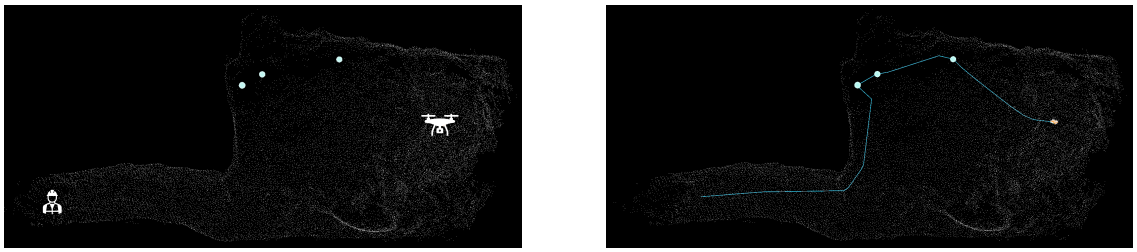


(a) (b)

Figure 1.11 Objective 3 - Semi-autonomous navigation. Once the area to be explored is identified by the pilot, with the solid gray line in (a), the objective is to find the shortest path to get there, represented by the solid black line in (b).

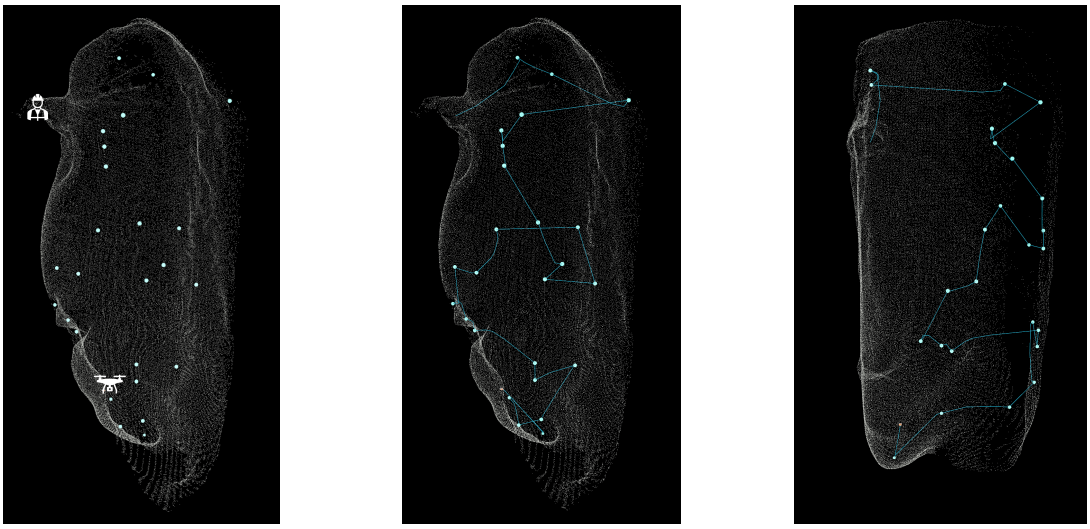
1.4.4 Autonomous exploration

Finally, regarding the entire mission, human beings do not have good intuition to optimize a global route when the problem, similar to the *Traveling Salesman Problem* (TSP) [Worboys, 1986], becomes large, especially in 3D. The problem is to find the optimal path to go through different points of interest and finish at a specific point. Fig. 1.12 shows an example for which a human performs well when the problem is small and similar to a 2D problem. Fig. 1.13 provides an example for which the human has more difficulty in finding the optimal solution, as the problem is large and difficult to visualize [Macgregor and Ormerod, 1996, Dry et al., 2006]. The fourth objective (see Fig. 1.14) addresses the question *"Where to go first?"* and is to autonomously choose the order of visit of the areas to explore.



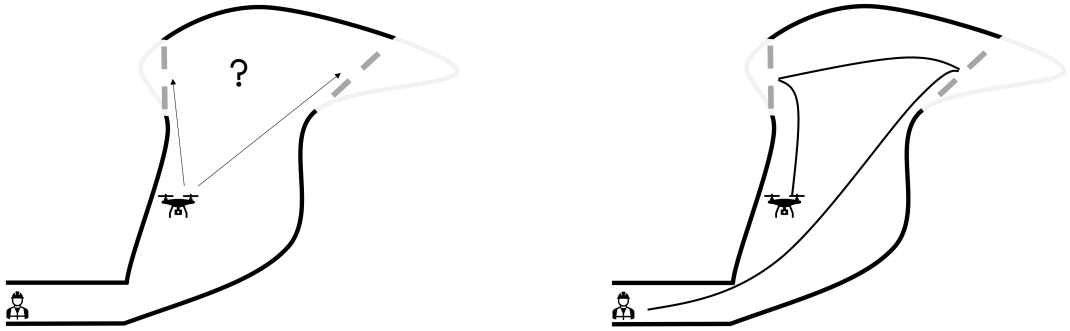
(a) (b)

Figure 1.12 Visualization of the best solution to an easy instance of a TSP: (a) problem instance with the drone on the right, 3 points to connect, and the stope entrance on the left, (b) solution.



(a) (b) (c)

Figure 1.13 3D visualization of the best solution to a hard instance of a TSP: (a) problem instance with the drone on the bottom, 27 points to connect, and the stope entrance on the top, (b, c) optimal solution from different perspectives.



(a) (b)

Figure 1.14 Objective 4 - Autonomous exploration. The final objective is to relieve the operator of the mental load of selecting the areas to explore. The path represented by the black line in (b) is the optimal sequence for visiting these areas.

Put together, these 4 objectives lead to an autonomous exploration algorithm. Regarding software only, the goal is to reach an autonomy level 5 according to the SAE standards [1] for autonomous vehicles [SAE, 2018].

Autonomy An effective mine mapping drone must rely on the power of autonomy for several reasons. Generally speaking, the economics of manually piloted drones do not scale well as they are difficult to fly, easy to crash, and almost all of the operating budget goes to pilot training and wages [Emesent, 2019]. An autonomous mapping drone works in three phases: 1) observe the environment, 2) understand it by interpreting the constructed map, 3) act accordingly. A human pilot is not as efficient as a computer during these 3 phases. Regarding phase 1, the human is not able to perceive a 360° environment at a given time whereas the machine can easily process a 3D point cloud. Regarding phase 2, for the same reason, the human is not very efficient to understand the environment for a drone that navigates out of sight. The operator only has a vision of a virtual environment, which does not easily allow to perform the objectives 1 and 2 above. Without the visual aids, which are developed in this work, the operator has little idea of his starting point and of the unmapped areas when the environment becomes large. Regarding phase 3, the constraints of the project make it not easy to fly a drone and meet objectives 3 and 4 above by choosing a short local path, while paying attention to the tether, and the length of the anticipated global path.

1. "The vehicle is driven autonomously in all conditions." The complete system can reach, at best, a level 3 autonomy according to EarthSense standards for field robots ("the robot is capable of dealing with edge cases but the human still needs to be on the field to swap batteries, perform repairs, or rescue a stranded robot") since an operator is still needed for the mechanical maintenance of the drone and for the setup [EarthSense, 2022].

One could argue that the tether loses its interest by moving towards a fully autonomous solution as the main reason for a tether is teleoperation. Nevertheless, the sharing of calculation and the unlimited flight time is still interesting. In addition to the possibility of responding to a corner case error, the communication with the pilot remains essential for an autonomous flight because the geologists need to see the images of the onboard camera in real time, e.g., to identify the spatial configuration of the gold vein.

1.5 Original contributions

The first algorithm, *RRT-Rope*, is presented as a generalization of path planning for large-scale uncluttered 3D environments [Petit and Lussier Desbiens, 2021]. It aims at dealing with the size of the environments which increases the computation time of a path, and the trade-off which must be made between computation time and path quality, to minimize the mission time. The approach incorporates a deterministic method for shortening the first path found by a modified *Rapidly-exploring Random Tree* (RRT). In uncluttered environments, the paper shows that, due to the topology of the environment, this method provides a near-optimal path. Our algorithm finds this path in a much shorter time (between 30% and 70% faster) than the state-of-the-art next best algorithm, in large-scale environments such as mining stopes. In addition to better accuracy, the results also show a better precision on the path cost obtained (smaller variance in solution quality).

The second algorithm, *TAPE*, is presented as a generalization of an exploration algorithm for unknown cavities with a tethered drone [Petit and Lussier Desbiens, 2022]. It aims at minimizing the mission time during the exploration by minimizing the path length, as well as ensuring the completeness of the map through the success of the mission which is ensured by a good tether management. This method is based on a two-level hierarchical architecture: a) the global planning solves a *Traveling Salesman Problem* (TSP) to minimize the exploration path length, and b) the local planning generates informed path options and chooses one of them according to a decision function that weights the path length and the tether unwinding. The paper shows that the overall exploration path has little to no influence on tether entanglements. On average, our method generates a 4.1% increase in distance traveled compared to the classic TSP solution, with which the length of the tether remains below the maximum allowed value in 53% of the 32 simulated cases against 100% with our method.

The two new algorithms developed are then used as technical tools to perform the tasks that belong to the scope of this project. These methods are combined with some additional techniques (e.g., filtering, path following, user interaction) to provide an intelligent and

robust tool for autonomous and fast exploration of underground mining stopes with a tethered drone. The system can also be used in interaction with a human operator to perform a RTH, provide an exploration assistance or navigate semi-autonomously to user-selected waypoints.

1.6 Document structure

First, Chapter 2 presents a literature review of existing methods for performing path planning and autonomous exploration, and provides a critique related to the problem stated here. It also presents the implementation of the state-of-the-art algorithms to solve the TSP which is presented in the TAPE algorithm. Then, Chapter 3 contains the paper that introduces the RRT-Rope algorithm—that allows to quickly compute a near-optimal path in large-scale environments. Chapter 4 features the paper that presents the TAPE algorithm—that leverages RRT-Rope and enables autonomous exploration of unknown cavities while taking into account the tether length and the global path length. Chapter 5 presents the integration of the techniques presented in the 3 preceding chapters into an intelligent system. Chapter 6 discusses the limitations of the presented methods and introduces development perspectives. Finally, Chapter 8 (Chapter 7 in French) concludes the thesis by highlighting the contributions made to the scientific community and to the industrial project by the RRT-Rope and TAPE algorithms.

CHAPTER 2

STATE OF THE ART

Advances in mobile robotics have led to a significant number of navigation algorithms developed in the last 30 years. This work focuses on the cognitive level of a robot, i.e., as defined in [Siegwart and Nourbakhsh, 2004], the purposeful decision-making and execution that a system utilizes to achieve its highest-order goals. This chapter describes state-of-the-art path planning techniques that can be used for the objectives described in Sections 1.4.1 and 1.4.3. Existing techniques for exploration are then described for the objectives described in Sections 1.4.2 and 1.4.4. Finally, existing algorithms for solving the TSP are compared in relation to the objective of Section 1.4.4.

2.1 Path planning

Path planning, one of the highest level aspects in robotics, concerns a set of algorithms that generate trajectories while respecting collision avoidance. The main challenge of the path planning problem is that it is difficult to compute the free space and occupied space, and thus capture the connectivity of the free space. Several strategies have been developed to try to represent the space through different types of approximations while trying to quickly find an optimal path. The complexity of the 3D path planning problem has led to many works [Yang et al., 2014].

As mentioned earlier, a mapping drone is likely to travel at low velocities. As detailed in Section 3.2.1, the path planning problem for such a drone can then be considered as a holonomic point-like problem for 3 reasons: 1) the approximation that the roll (ϕ) and pitch (θ) angles are zero (horizontal attitude) is valid, 2) the control of the yaw angle (ψ) can be decoupled from the control in position, 3) the drone can move in all directions.

The most popular families of techniques for a holonomic point-like path planning problem are gradient-based methods, node-based algorithms, and sampling-based algorithms. [1] These are detailed in the following sections.

1. Trajectory optimization methods (e.g., linear programming methods such as shooting or collocation) are not considered because they require a convex envelope—which is costly in 3D and impossible in a cavity—and the time to solve a non-collision with a distance equation is much more than a boolean collision check performed by searching an octree. Moreover, these methods do not take advantage of the triangular inequality, or the width of the reachable space, which give a good preliminary knowledge of the shape of the solution.

2.1.1 Gradient-based methods

One of the main ideas behind gradient-based methods is the use of Artificial Potential Fields (APF) [Koren and Borenstein, 1991]. These techniques are inspired by natural potential energy fields such as magnetic fields. The target represents a point of low potential, attracting the robot, and the obstacles are characterized by a high potential, exerting a repulsive force, as illustrated in Fig. 2.1.

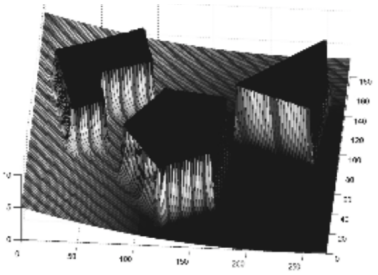


Figure 2.1 Virtual potential fields [Siegwart and Nourbakhsh, 2004].

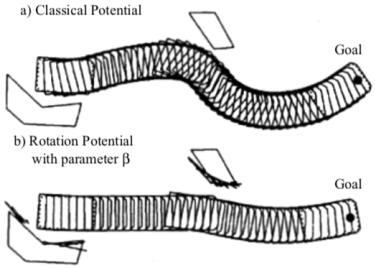


Figure 2.2 Extended virtual potential fields [Siegwart and Nourbakhsh, 2004].

These methods are quick to execute and can be performed in real time to give a dynamic response to the system. The main disadvantages of these methods are the local minima and the oscillations caused by the potential differences. To overcome these oscillations, it is possible to use an extended potential field, linked to the angle of the robot and taking little account of the obstacle if the trajectory is parallel to its wall, as presented in Fig. 2.2. It is also possible to add a damping factor to the force experienced by the robot. Nevertheless, since the walls of the stope can be very uneven, simulations confirmed that this method is not viable for autonomous navigation and the drone gets stuck during the mission [Dozois, 2022]. As mentioned in [Siegwart and Nourbakhsh, 2004], gradient methods are more for obstacle avoidance (i.e., reaction to the environment) than for planning, and this reacting effort cannot guide the overall robot behavior to reach a distant goal. Regarding the project covered in this study, the potential field is used as an obstacle avoidance system for pilot assistance, as detailed in [Dozois, 2022]. With this system, the pilot is able to use the joysticks of the remote control as an attractive field and the local obstacles are used to compute a repulsive field, allowing to assist the pilot in his mission for a good avoidance of the stope walls. Also, the human can easily get the robot out of the local minima.

2.1.2 Node-based algorithms

Node-based methods rely on the discretization of the reachable space, separating free cells and cells occupied by obstacles. In 2D, nodes can be connected to 4 or 8 neighbors, allowing diagonal movements, as seen in Fig. 2.3 and 2.4. For 3D spaces, by analogy, they

can be connected by 6, by 18 for 2D diagonals or by 26 for 3D diagonals. The cost to pass in a diagonal must still be penalized, being further away than a neighboring element along a single axis. This penalty can be done using the Euclidean distance, illustrated in Fig. 2.5. The Manhattan distance, also represented in Fig. 2.5, should be used when the cells are connected by 4 in 2D, or by 6 in 3D. Variations of the node-based methods are presented in this subsection.

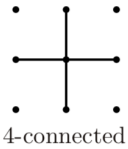


Figure 2.3 Grid by 4 [Lynch and Park, 2017].

Figure 2.4 Grid by 8 [Lynch and Park, 2017].

Figure 2.5 Distance metrics in a grid [Lynch and Park, 2017].

Exact methods The exact methods are based on the geometry of the obstacles and the space. This means that the cells are either completely free or completely occupied, as illustrated in Fig. 2.6. In 3D, these methods are relatively complex to implement in real-time. Also, the size of the cells depends on the complexity of the environment, so the movement of the robot can be unpredictable, especially if the cell is large.

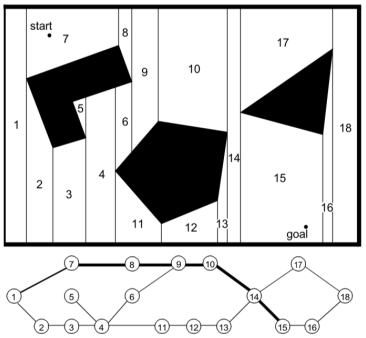


Figure 2.6 Exact method grid [Siegwart and Nourbakhsh, 2004].

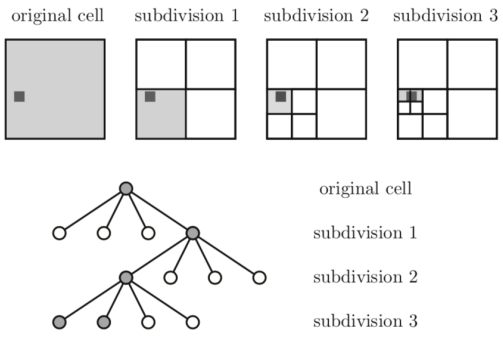


Figure 2.7 Multi-resolution grid [Lynch and Park, 2017].

Approximate methods Approximate methods, which are much more popular, are based on a fixed cell size (except in the case of multi-resolution grids as in Fig. 2.7). This means that the cell size does not depend on the geometry of the environment, which is a disadvantage in the case of complex obstacles, i.e., some narrow passages would not be marked as free. Generally, the mesh size is fine enough to avoid this problem. Among these techniques, the $A*$ algorithm [Hart et al., 1968] is popular due to its low

algorithmic complexity, compared to, e.g., *Dijkstra* [Dijkstra, 1959] which evaluates every node regardless of the goal direction. The major advantage is that it is complete in resolution, i.e., if a path exists and the grid resolution is fine enough, the path will be found. Moreover, this path will be the shortest.

A* algorithm At each iteration, A* minimizes a function

$$f(n) = g(n) + h(n)$$

to choose the next node n (represented in Fig. 2.8). The function $g(n)$ is the cost of the path from the initial point (q_I) to the node n while $h(n)$ is the heuristic that estimates the cost of the least expensive path from n to the target (q_G). The algorithm analyzes different paths in parallel and these paths are dropped if the value of f is higher than all other paths considered. The algorithm terminates when one of the paths starts from the initial point and reaches the target or when no more paths are admissible. This algorithm is used for many practical applications but the number of cells increases exponentially with the size of the problem [2]. The complexity therefore increases rapidly in 3D due to the number of node connections mentioned at the beginning of Section 2.1.2.

(a) (b)

Figure 2.8 A* algorithm: (a) after one iteration, (b) final path.

A* time complexity The complexity of A* is $\mathcal{O}(b^d)$ where d is the solution depth, which depends on the distance to travel between the starting and ending point, and on the refinement of the grid. The term b represents the branching factor, i.e., the average number of successor nodes of a same node. It depends on the heuristic and the dimension of the graph. The optimal heuristic, i.e., the true distance to travel, gives b=1, regardless of the number of dimensions. In 1D, the worst heuristic gives $b = 2$. In 2D, $b = 4$ and in 3D, $b = 6$, without diagonal displacements, and the algorithm thus becomes equivalent to Dijkstra.

2. Here, the term refers to several metrics: the spatial size, the grid resolution and the size of the parameter space.

In practice, for a simple path in an open 2D space (e.g., 10 m solution in 2D, with a mesh size of 0.5 m and b = 1.2), the grid contains 400 nodes for a square bounded space and A* would perform around 40 operations to find a path. For a path planning problem in a complex and large stope (e.g., 50 m solution in 3D, a mesh size of 0.2 m and b = 2), the grid contains $15 * 10^6$ nodes that must be stored in memory for a cubic bounded space. In this grid, A* would perform around $2 * 10^{75}$ operations. To give an order of magnitude, this is almost the number of atoms in the universe. As stated in [Hauser, 2022], techniques such as A* are optimal and complete in resolution, but their performance depends heavily on the grid resolution, and somewhat on the grid search heuristic. This type of planning is time consuming in large-scale environments such as these considered in this work.

2.1.3 Sampling-based algorithms

For our application, potential field methods lead to local minima problems. On the other hand, node-based techniques produce optimal paths but their algorithmic complexity for large problems makes them difficult to implement in real time in large-scale 3D environments. Another class of methods, called the sampling methods, give satisfactory solutions more quickly in large-scale environments. These methods aim at selecting a sample of the reachable space. They are based on a first step that evaluates if a sample is located in the obstacle-free space, then a second step determining the closest sample among those reachable, and finally a local planner connecting them. These steps are then repeated. The samples, in smaller number than with a decomposition grid, form a roadmap or search tree. This allows to find a solution in a statistically complete way [3]. Among these methods, search trees are the most used, especially the *Rapidly-exploring Random Trees* (RRT) [LaValle and Kuffner, 2000, Véras et al., 2019]. The most useful variants are described below.

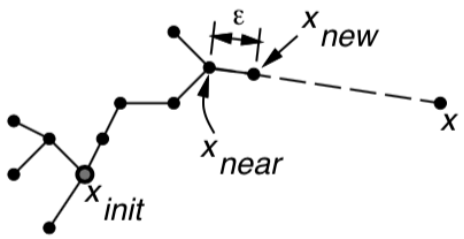


Figure 2.9 Extension procedure of an RRT [Lindemann and LaValle, 2001].

3. The probability of finding a solution, if it exists, tends to 1 for a number of samples approaching infinity.

RRT These trees are created using the generation of random points called x_{samp} in the C-space (configuration space). The points are generated according to an almost uniform distribution with a bias towards the target. The node x_{near} of the tree closest to the random point is the node minimizing the heuristic to get there, as represented in Fig. 2.9. A new point x_{new} is thus created, at a predefined distance ϵ from x_{near} on a straight line between it and x_{samp}. If the straight line between x_{near} and x_{new} is entirely contained in C_{free}, the point x_{new} is added to the tree. There are a large number of variants of this algorithm, varying the way x_{samp} are generated [Gammell et al., 2014], varying the heuristic for selecting x_{near} for a given x_{samp} [Shkolnik et al., 2009], or even varying the path planning between x_{near} and x_{new} which can be more complicated than a straight line [Hu et al., 2021]. The difference between the RRT briefly described here and a random tree where each point would be randomly added around an already existing node is highlighted in Fig 2.10. The classical random tree, displayed on the left side of Fig 2.10, is biased towards the already explored locations while RRT, displayed on the right side of Fig 2.10, has a Voronoi bias. This bias is shown in Fig. 2.11. In this figure, the Voronoi regions related to the nodes of the tree (formed by the dark blue segments) are separated by the red lines. The nodes at the end of the branches occupy the largest Voronoi regions since the space beyond is unexplored. Thanks to the random generation of x_{samp}, the x_{new} tend to occupy these large Voronoi regions and thus reduce their size. The diffusion of the graph is therefore uniform within the C-space.

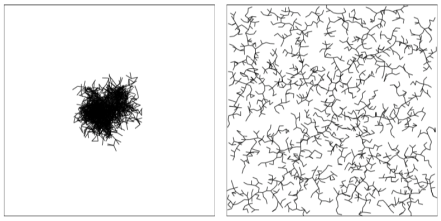


Figure 2.10 Classical random tree vs. an RRT [Lindemann and LaValle, 2001].

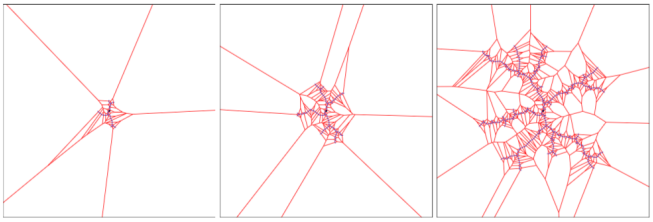


Figure 2.11 Illustration of the Voronoi bias for RRTs [Lindemann and LaValle, 2001].

For a path planning answering a holonomic point-like problem, it is more relevant to use $x_{new} = x_{samp}$. Indeed, in a holonomic problem, the probability that a random node is reachable is higher, thanks to the absence of dynamic constraints. This is even more true in an uncluttered environment such as a stope. There will be no infinitesimal step ϵ to take into account, which gives a different aspect to the tree [Lindemann and LaValle, 2001].

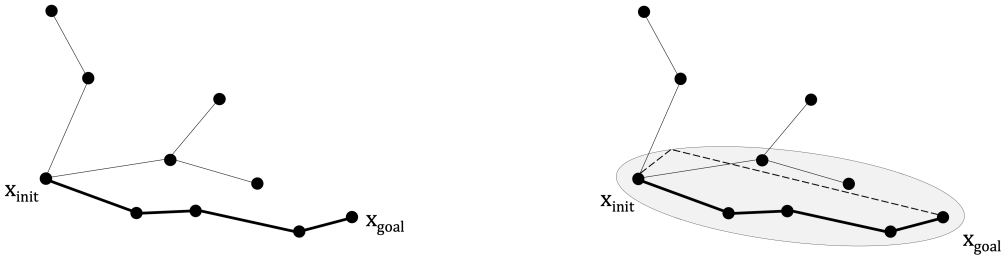
RRT* RRT algorithms are popular because of their probabilistic completeness and quick path computation in single-query problems, but they fail to converge to optimal solutions because of existing graph bias [4]. *RRT** is an evolution of RRT that ensures asymptotic optimality [Karaman and Frazzoli, 2011]. With this algorithm, each new node in the tree is the center of an inspection in a given radius around it, as represented in Fig. 2.12. If, within this radius, a node has a cost greater than the cost it would have if the new node were its parent node, these nodes are reconnected by a new branch. This algorithm is generally effective in converging faster to the optimal path in fewer iterations, but each iteration becomes longer to compute, depending on the size of the chosen radius.

(a) (b)

Figure 2.12 RRT* algorithm: (a) after adding x_{new} to the tree, (b) after rewiring.

Informed RRT* Once a first path has been found, a random point generated outside a certain region will never lead to a shorter path than the one already found. In 2D, this region is an ellipse, having for foci the start point and the goal point, and for major axis the length of the found path. In the algorithm called *Informed RRT** [Gammell et al., 2014], random points are only generated inside this ellipse (or ellipsoid in 3D) in order to quickly converge to a more optimal path. Fig. 2.13 illustrates this algorithm. Let us consider a path with only 1 node between the start and the goal point. If the node is located on the curve of the ellipse, the path will be of the same length as the major axis of the ellipse. This is illustrated by the dotted path. This technique allows better convergence rate and solution quality than RRT* but it requires that a first path is found by RRT* or RRT. The computation of a first path is expensive if the goal position is challenging to reach.

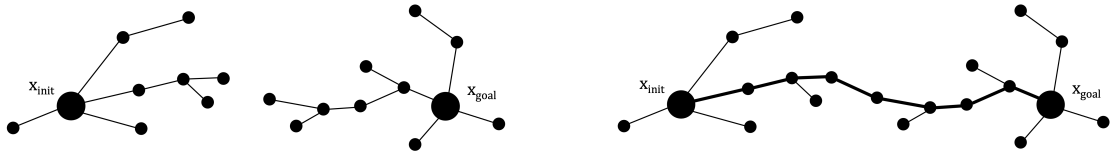
4. The final path is strongly biased towards the existing graph due to the connection step with the closest node of the tree.



(a) (b)

Figure 2.13 Informed RRT* algorithm: (a) first path found, (b) random point generation ellipse (gray).

RRT-connect In order to find a first path between 2 points faster, *RRT-connect* [Kuffner and LaValle, 2000] deploys 2 trees, one at the starting point and the other at the goal point. These trees, shown in Fig. 2.14, are intended to join each other and the generation of random nodes is biased towards the last node added to the other tree. This method is very effective for fast path finding but is not optimal.



(a) (b)

Figure 2.14 RRT-connect algorithm: (a) before, (b) after finding a path.

Optimization techniques Some ideas have been developed for online path optimization, including RRT# [Arslan and Tsiotras, 2013], C-FOREST [Otte and Correll, 2013], searching around beacons [Kanehara et al., 2007], and graph pruning [Karaman et al., 2011, Gammell et al., 2018]. To improve computation time through data management, graph pruning techniques remove some vertices according to a chosen heuristic, e.g., the sum of the node cost and the distance from node to goal [Karaman et al., 2011], or the sum of the distance to start and distance to goal [Gammell et al., 2018]. The latter is used in Informed RRT*-connect [Mashayekhi et al., 2020], an algorithm combining the advantages of all the algorithms introduced so far, as presented in Fig. 2.15.

Although graph pruning improves computation time, it still needs considerable time to compute an optimal path in large-scale 3D environments, due to the asymptotic nature of RRT variants optimality. Fig. 2.16 shows how RRT, RRT* and Informed RRT* perform in an average stope, with the same computer as described in Chapters 3 and 4. The curves asymptotically approach the optimum and the convergence rate is very slow.

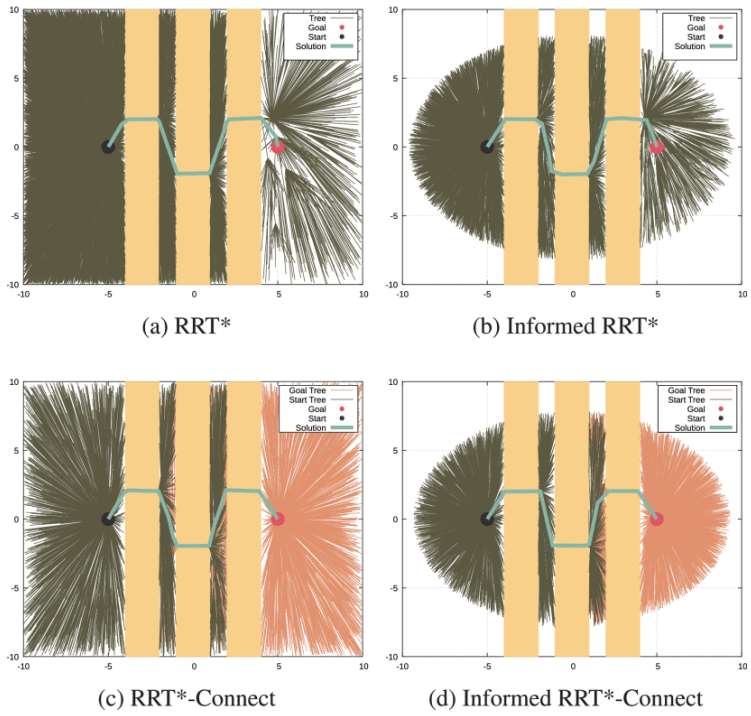


Figure 2.15 Example of a path request with Informed RRT*-connect [Mashayekhi et al., 2020]. The obstacles are represented by the yellow columns which are cut in their center by a small opening. The path found by each method is represented in blue.

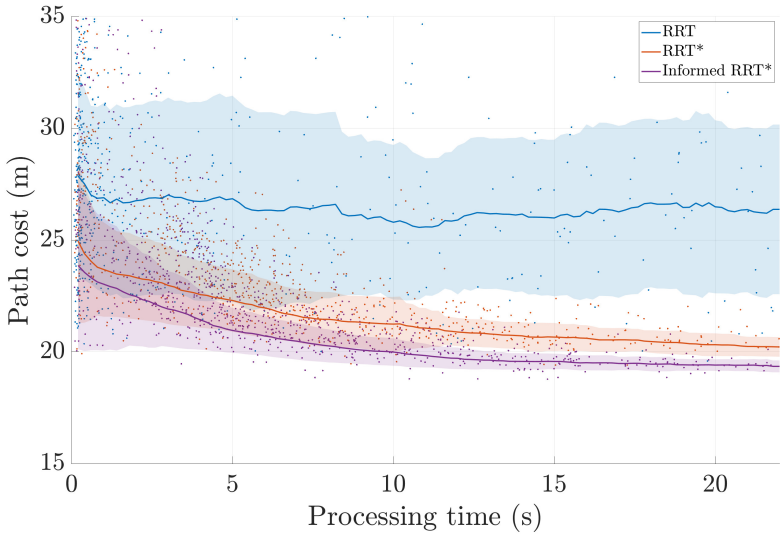


Figure 2.16 Moving average of path cost over a 7-second window for RRT variants in an underground mining stope. Several path queries were computed between 2 points located at the ends of a representative stope. The path between these 2 points typically corresponds to the case of an RTH after an exploration mission. The shaded region corresponds to the moving average of the standard deviation.

Post-processing techniques Post-processing techniques have been introduced to overcome the problem of asymptotic optimality. The performance of path optimization is largely dependent on the environment topology and size. Path optimization generally takes longer to calculate in large environments and is not very efficient if the environment is very cluttered. Underground mining stopes have a high probability of being 3-ball homotopic [5] by the nature of their creation by explosion. The surface of a stope is a connected space, as defined in Fig. 2.17, and the volume of the stope is simply connected [6] because there are no suspended obstacles, as pictured in Fig. 2.18.

(a) (b)

Figure 2.17 Topological spaces proprieties: (a) not connected vs. (b) connected.

(a) (b)

Figure 2.18 Topological representation of an underground mining stope: (a) admissible stope, (b) not a stope.

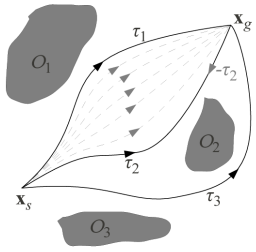


Figure 2.19 Representation of homotopy classes [Bhattacharya et al., 2012].

Fig. 2.19 shows an example of homotopic paths. τ_1 is homotopic to τ_2 but τ_3 belongs to another homotopy class, i.e., it cannot be continuously deformed into any of the other two. As [Luna et al., 2013] stated, shortcutting is unlikely to discover a new homotopy. However,

5. Two spaces that have the same "shape", from a topological point of view, are homotopic, i.e., they can be continuously deformed into the other. The 3-ball homotopy group gathers the spaces that are homotopic to the interior space of a 2-sphere (a sphere in 3 dimensions).

6. Any loop can be reduced to a point without encompassing any domain not included in the space.

in a 3-ball homotopic environment, any path is, by definition, in the optimal homotopy class. The idea of shortcutting is therefore very interesting in these environments.

Shortcutting an RRT is an old idea that has been proven to outperform optimal RRT variants [Luo and Hauser, 2014]. As demonstrated later in Section 3.5, with an efficient shortening method, the final path cost does not depend on initial path quality, and online optimization is therefore obsolete in stopes. Different techniques use the triangular inequality principle to prune some nodes of a graph or a path to shorten it, as illustrated in Fig. 2.20. The principle of triangular inequality is based on the fact that any path $A \Rightarrow C$ that can be made of a straight line, without colliding with an obstacle, will be shorter than $A \Rightarrow B \Rightarrow C$.

(a) (b)

Figure 2.20 Path optimization algorithm by node pruning: (a) before, (b) after pruning.

This node pruning technique is implemented in [Kanehara et al., 2007, Islam et al., 2012]. Fig. 2.21b shows that the resulting path is shorter than the raw path but not optimal because the node resolution is reduced through node pruning.

Some iterative techniques have been developed to try to converge to an optimal path [Berchtold and Glavina, 1994, Brock and Khatib, 2002]. Elastic strips, depicted in Fig. 2.21a, is a method that locally displaces a point according to a virtual attractive force towards its two neighbors and a repulsive force coming from obstacles. Since the algorithm is designed for online control purposes, the convergence is slow when used as a shortening method, even with accelerated gradient descent techniques.

(a) (b) (c)

Figure 2.21 Examples of path post-processing techniques: (a) elastic strips (from [Brock and Khatib, 2002]), (b) node pruning (from [Islam et al., 2012]), (c) partial-shortcut (from [Geraerts and Overmars, 2004]).

The most popular path shortening approach is partial-shortcut [Geraerts and Overmars, 2004]. It selects 2 arbitrary points in a path and moves the path segment between them on a straight line (or linear interpolation for multiple DOF systems) if there is no collision. Since points are moved instead of pruned, the path is closer to optimality, as illustrated in Fig. 2.21c. However, computation time is not optimal due to some irrelevant shortcuts performed because of the non-deterministic point selection. This can happen if the selected portion of path is already a straight line segment, or if this portion is meant to be pruned by another shortcut in a future instance.

2.1.4 Preliminary conclusions

A comparison study [Zammit and Van Kampen, 2021] concluded that A* can outperform RRTs in complex and cluttered environments in terms of path finding time and path quality. Nevertheless, as demonstrated in Section 3.5, RRTs perform well above A* in large-scale uncluttered environments. As stated by [Hauser, 2022], sampling-based motion planners can overcome some limitations of the curse of dimensionality compared to grid search methods, but they pay a cost in the variance of solution quality and running time. This running time depends on the visibility characteristics of the free space, which is fast in spaces of good visibility, such as these described in this study.

Finding a feasible path is generally very efficient with RRT-like techniques such as RRT-connect, but online optimization remains time-consuming in large-scale environments. In such environments, most shortcutting methods require many iterations. In addition, the algorithm usually stops when the computation time exceeds a value set by the user, or when the measured convergence rate is sufficiently low. The performance is therefore variable and unpredictable.

An efficient path planning algorithm must leverage the knowledge of the problem to provide an accurate solution faster, and with good precision. As said earlier, the problem is holonomic and point-like, and the environment is uncluttered and large-scale. For these reasons, there is little chance of encountering collisions. With this in mind, it is possible to modify RRT-connect to generate a first feasible path very quickly. Also, since any path is homotopic to the optimal path, efficient shortcutting will quickly generate a near-optimal path. To summarize, the objective is to decrease the computation time and the variance of the path quality. The remaining question is how to find a mapping between the first path τ_f, in yellow in Fig. 2.22, and the optimal path τ^* in blue. The main challenge is to find a good approximation to capture the connectivity of the space. This idea is developed in Chapter 3.

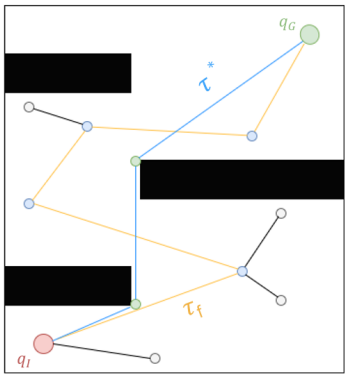


Figure 2.22 Representation of the path shortening problem [Petit and Lussier Desbiens, 2021].

2.2 Exploration

The literature review in the previous section focused on the path planning problem, which answers the question of how to move around obstacles. Attention now turns to to the exploration problem, which addresses the question of where to go to increase map knowledge. This problem arises in robotic mapping [Papachristos et al., 2019] and in Search And Rescue operations [Balta et al., 2016, Hong et al., 2019]. A lot of work has been done to tackle this problem in order to allow robots to navigate in dangerous environments or inaccessible to humans, without prior knowledge of the map [Lluvia et al., 2021]. This section presents different exploration methods, details the frontier detection and selection methods, and then discusses techniques that consider the case of a tethered robot.

2.2.1 Exploration techniques

Exploration is defined as:

> "[...] the act of moving through an unknown environment while building a map that can be used for subsequent navigation." [Yamauchi, 1997]

This is most formally described by the *Wake-up Robot Problem* [Fox and Burgard, 1999], in which an autonomous robot is put into operation at an unknown location without any preliminary knowledge of the space. The most successful families of techniques for the exploration problem are topological methods, information theory, random sampling-based methods, and frontier-based methods [7]. This classification is theoretical, and several recent methods combine 2 or 3 of these techniques. The details are presented below.

Topological methods Topological exploration is based on the topological representation of the environment [Thrun, 1998, Kim et al., 2013, Silver et al., 2006b, Silver et al., 2006a]. This can be done with a connectivity graph based on map segmentation, which is represented in Fig. 2.23a. These methods are closely related to localization and mapping through loop closure detection, but they can help in decision making for exploration. The decision making method is based on the construction of a graph containing the detected nodes or edges. The graph prevents from exploring the corridors already explored, during an intersection. With edge and intersection detection, as shown in Fig. 2.23b, the method is efficient in tunnel networks or offices by focusing on topological completeness. Fig. 2.24 shows an example in a subterranean tunnel network. The robot must choose between the left and right corridors which are detected by the green dots. Topological methods are not suitable for large cavities such as stopes, which often cannot be divided into separate

7. Machine learning techniques are not considered because the project does not provide the possibility to capture enough data to train a model with only a dozen mapped environments. One approach that could be considered would be data augmentation of maps but this avenue has not been explored.

sections. A good stope exploration algorithm requires map completeness (e.g., for volume calculation, as explained in Chapter 1), whereas these methods only guarantee topological completeness, i.e., that each branch or room of the network is visited.

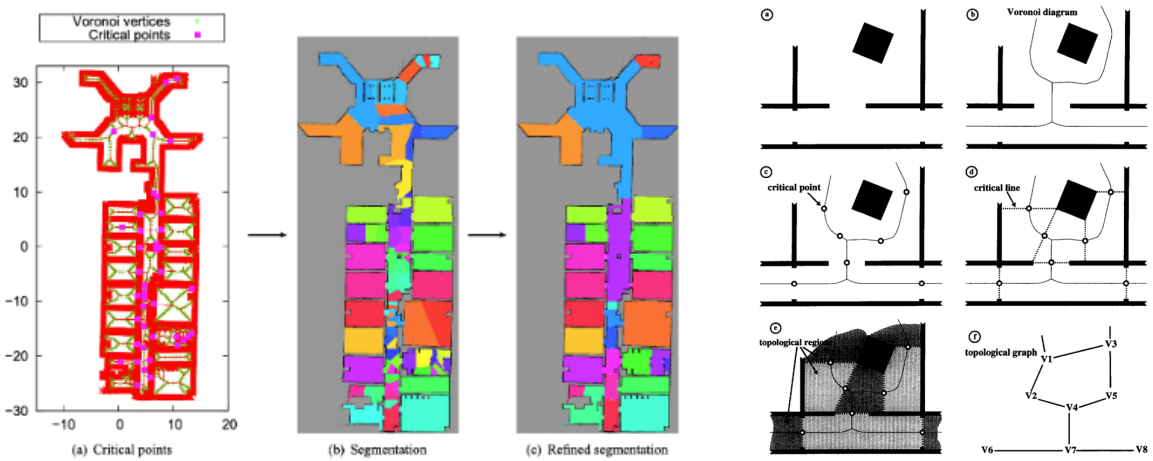


Figure 2.23 Tools for topological exploration: (a) map segmentation for an office-like environment (from [Holz et al., 2010]), (b) edge and intersection detection (from [Thrun, 1998]).

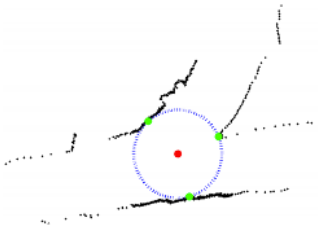


Figure 2.24 Example of a decision making situation using topological exploration methods [Silver et al., 2006a].

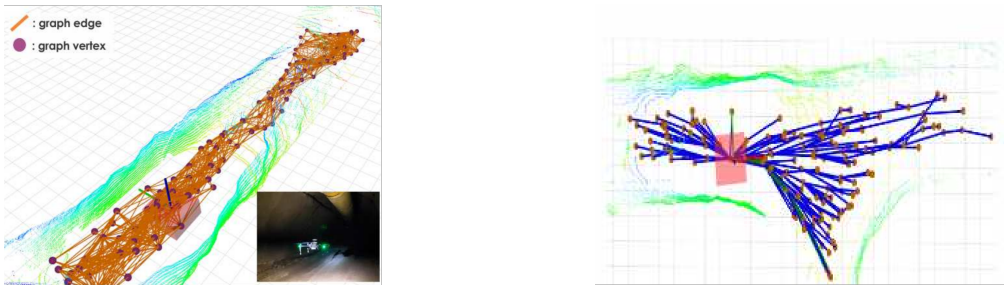
Information theory With approaches based on information theory, the objective of the robot is to maximize the amount of new information to expect for the next sensors readings. For this, the robot evaluates samples of hypothetical next positions in the near future, as shown in Fig. 2.25b, and estimates the gain in map knowledge through metrics depending on the methods used (e.g., maximizing the number of new cells discovered, or minimizing uncertainty) [Bai et al., 2016, Tabib et al., 2016, Stachniss et al., 2005, Shade, 2011, Vanegas et al., 2019]. Fig. 2.25 shows an example where cells are characterized by a Shannon entropy measure. Visited cells do not contribute to the entropy and unvisited cells contribute the most. The best actions determined with these methods are often very

local and do not consider the anticipated global path. Information theory is thus often greedy and global optimality is affected by their myopia.

(a) (b)

Figure 2.25 Example of an exploration mission based on information gain as described in [Bai et al., 2016]: (a) acquisition function with the maximum in red, (b) selection of the next pose samples. The robot is in red, the evaluated samples in blue, and the next best actions in pink.

Random sampling-based methods Random sampling-based methods can be seen as an evolution of information theory in a more distant mission horizon. They are based on growing an RRT [LaValle and Kuffner, 2000] or a similar planner [Karaman and Frazzoli, 2011]. Most of random sampling-based methods integrate information theory in order to choose the most promising branch [Papachristos et al., 2019, Dang et al., 2018]. Among these, Next-Best-View Planner (NBVP) [Bircher et al., 2016] and Graph-Based exploration Planner (GBP) [Dang et al., 2019] are considered among the most efficient state-of-the-art exploration algorithms. They are based on deploying a random graph (shown in Fig. 2.26a) and then evaluating the amount of space that can be mapped for the different poses of this branch, as pictured in Fig. 2.26b. Then, the first branch step is performed and the tree is recomputed with the new map knowledge, in a receding horizon fashion. They are very effective in tunnel-like environments but often suffer from a) randomness and therefore neglect of partially explored areas, or b) greedy behavior when a reward gain is assigned, in the same manner as information theory.



(a) (b)

Figure 2.26 Example of an exploration mission based on GBP [Dang et al., 2020]: (a) local graph building, (b) exploration gain evaluation.

Frontier-based methods The last method is based on the notion of *frontiers* [Yamauchi, 1997]. This is the most widely used method in the literature [Ravankar et al., 2018, Bachrach et al., 2010, Shade, 2011, Dornhege and Kleiner, 2011, Gao et al., 2018]. Fig. 2.27 shows how they can be detected in a 2D map. The frontiers correspond to the cells located between the free space and the unknown space. In the same figure, zones 0 and 1 correspond to open doorways, and zone 2 corresponds to the end of the unexplored hallway. This technique is based on a process analogous to edge detection [Canny, 1986] and region extraction [Ito et al., 1996] in computer vision, which is very fast to compute from a 2D image.

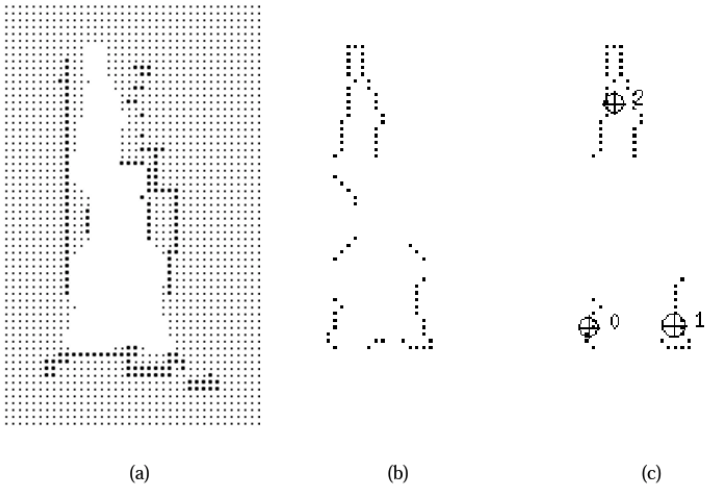


Figure 2.27 Representation of frontiers: (a) evidence grid with the free space in white, unknown space in gray and occupied space in black, (b) frontier edge segments, (c) frontier regions [Yamauchi, 1997].

For the robot to explore the environment, the method selects the nearest frontier and the map is expanded incrementally by moving the robot accordingly. Other frontier selection methods have been proposed since then. Some methods combine frontier and information-based methods [Shade, 2011, Shen et al., 2012]. Fig 2.28a shows the expected view from a hypothetical point of view in the reachable space. The method considers the number of frontiers that could be observed from this point and finds the point that maximizes that number [Freda et al., 2008]. Fig. 2.28b shows an extension of this method that minimizes a metric that weights the number of visible frontiers and the distance to the viewpoint [González-Baños and Latombe, 2002].

As a whole, the notion of frontiers is interesting to best represent the space that remains to be discovered. Nevertheless, the frontier selection techniques mentioned here are not appropriate to the stope mapping problem. The original paper states that the main

(a) (b)

Figure 2.28 Information-based frontier selection algorithms: (a) classic, (b) weighted. The viewpoint selected is shown in gray, and the frontiers visible within the sensor range are represented by a bold line of dashes.

question in robotics exploration should be *"Given what you know about the world, where should you move to gain as much new information as possible?"*. As explained in [Juliá et al., 2012], the appropriate algorithm depends on the problem concerned. The main objective of stope mapping is to cover the environment completely, in the most efficient way possible. A proper algorithm should focus on map completeness and path efficiency instead of the speed of discovering most of the map greedily, as many algorithms do.

2.2.2 3D frontiers detection

Since frontier-based exploration seems to be the most promising technique, frontier detection is further developed in this section, and frontier selection tools are proposed in the next section to optimize the global path. The original frontier detection algorithm is based on image computation but this approach is not applicable in 3D [Canny, 1986]. Some preliminary notions must be introduced to understand how to apply this principle in 3D. The first important notion is a *voxel*. Just as the word pixel comes from the prefix "-pix" (or pics) for "picture" and the suffix "-el" for "element", a voxel is an element of volume. A map can be built on the basis of these voxels which are each a cube of fixed side length and have an occupancy state. In the framework of an occupancy grid [Moravec and Elfes, 1985], the elements are linked to an occupancy probability p, initialized at p_0. Based on this value, the 3 possible occupancy states are *free* if no obstacle is present ($p < p_0$), *occupied* if it is an obstacle ($p > p_0$), or *unknown* if it has not yet been observed by the sensors ($p = p_0$). To accumulate and retrieve this data in large spaces, tree data structures have proven to be very effective.

An *octree* [Meagher, 1980] is a tree in which each parent voxel is divided into 8 child voxels, called octants. This structure helps to easily add data when a map is initially unknown and therefore unbounded. It also allows to quickly insert or find the value of a voxel at a

desired definition using classical tree search methods (such as Depth-First Search (DFS) or, better yet, methods that use the geometric properties of the octree [Castro et al., 2008]). An octree populated with an occupancy map is called an *OctoMap* [Hornung et al., 2013]. A major advantage of an octree, represented in Fig. 2.29 is that the resolution (i.e., tree depth) is not uniform depending on the mapped object or environment. This translates into more efficient memory management, using many points only where accuracy is required.

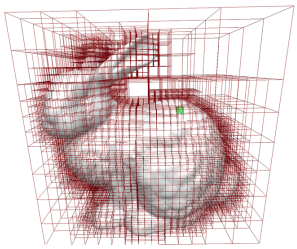
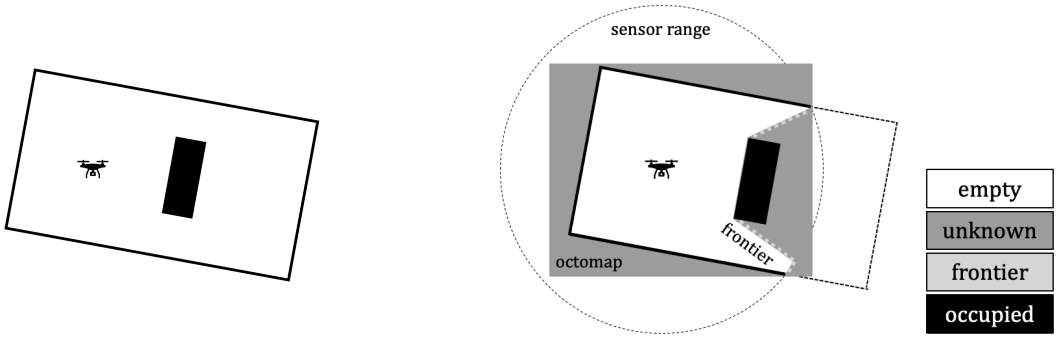


Figure 2.29 Octree representation [Castro et al., 2008].

Returning now to the exploration mission, the objective is to map the environment completely, i.e., to fill the space with free and occupied voxels, as in Fig. 2.30a. At a given time, the robot must observe the unknown voxels but not all of them are reachable. The frontiers are unknown voxels that are neighboring free cells, and thus visible from a point in the free space [8]. These are the unknown voxels that are either shadowed by obstacles or outside the range of the sensors.



(a) (b)

Figure 2.30 Illustration of frontiers between free and unknown space.

The naive method to detect frontiers is by searching the octree for neighboring unknown voxels for each free voxel [Hornung et al., 2013]. More effective approaches have been

8. The free space mentioned here is defined in the occupancy grid, and is not necessarily equivalent to the free space of the robot. The point is potentially reachable if it is in the free configuration space, e.g., if the width of the environment allows it.

proposed to enable real-time computing for an expanding map. The work in [Keidar and Kaminka, 2014] and [Quin et al., 2014] uses a process analog to a Breadth-First Search (BFS) in free voxels. Visited voxels are marked to focus on updated voxels only to avoid unnecessary calculation of the already identified voxels. Another technique has been designed to optimize the BFS by starting at ray endpoints in the sensor radius of the new scan [Quin et al., 2021]. The results are very effective and the time complexity of an iteration is in $\mathcal{O}(n)$ considering n frontiers.

2.2.3 Global path optimization

Now that the implementation techniques for 3D frontier detection have been detailed, this section focuses on frontier selection that leads to an optimal exploration path. An efficient method for the optimization of the exploration path is called Technologies for Autonomous Robot Exploration (TARE). It computes viewpoints similarly to the frontier-based methods and solves a TSP [Cao et al., 2021b, Cao et al., 2021a]. This method explores in a more complete and fast way than NBVP [Bircher et al., 2016] and GBP [Dang et al., 2019]. The effectiveness of TARE is partly due to the hierarchical architecture. GBP has also made an effort towards a hierarchical architecture. Concerning TARE, the local level stores data densely and computes a detailed path in a local planning horizon, while the global level maintains data sparsely and computes a sparse path at the global scale, as shown in Fig. 2.31. This makes real-time computing much easier.

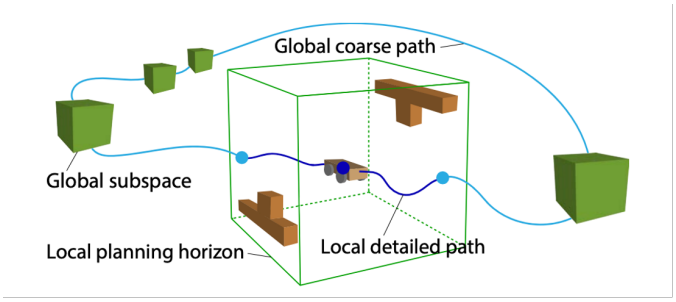
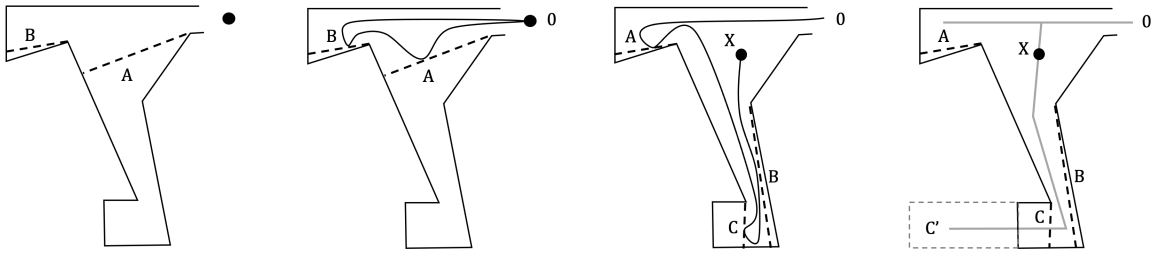


Figure 2.31 TARE exploration framework [Cao et al., 2021b].

The other factor that makes TARE effective is the TSP formulation. Some frontier-based approaches have also considered the selection problem as a variant of the TSP [Kulich et al., 2011, Kulich et al., 2019]. The Traveling Salesman (or Salesperson) Problem [Worboys, 1986] consists of finding the shortest route for a traveler who has to visit all of the cities in a list, where the distance between the cities are known.

Indeed, the frontier selection problem seems very similar to the TSP formulation. Let us consider an exploration where the drone has several frontiers to visit, as pictured in Fig. 2.32. A frontier leads either to a dead-end or to a new part of space containing

one or more frontiers. In the second case where the drone would continue to explore, as the volume of a stope is a simply connected space, any path between C' and the base station 0 automatically passes through C again. Thus, the distance $d(C, C')$ is added to the total path length, regardless of the order of visit chosen at time $t = 1$ between A, B and C. The optimal global solution based on the knowledge of the map at a given time is therefore very close to the real optimal global solution. By repeating the TSP computation incrementally with the latest map data, the drone decision thus approaches the desired optimal behavior. Basically, the frontier selection problem comes down to a TSP.



(a) t = 0. (b) t = 0 (with path). (c) t = 1 (with path). (d) t = 1 (simplified).

Figure 2.32 Similarities between the TSP and the frontier selection problem.

A minor disadvantage of TARE is that they do not consider the Return To Home in the optimization of the exploration path once the map is fully explored. In tunnel-like or office environments, this is of little importance since the entire path is composed of very similar round trips in narrow corridors. In large cavities, it is a significant factor to consider since the paths are less likely to overlap due to the width of the environment [9]. A proper definition of the TSP should therefore include the base station as a node in the graph.

State-of-the-art algorithms to solve the TSP are detailed in Section 2.3.

2.2.4 Tether consideration

The methods mentioned so far do not consider the case of a tethered drone. The next two sections mention exploration methods that address this case and existing techniques for modeling a tether. It is desirable to have a rewindable tether to help with the proper modeling and to avoid getting caught on the ground with a dangling tether. The tether is assumed to be always in taut condition. As said in Section 1, the extra tether should be stored in an onboard spool, allowing the drone to continue its mission even if the tether gets caught in obstacles. This setup is represented in Fig. 2.33 with the NetherDrone that

9. More info in Fig. 2.44.

fulfills these characteristics except for the rewind which is to be integrated in the future of the project.

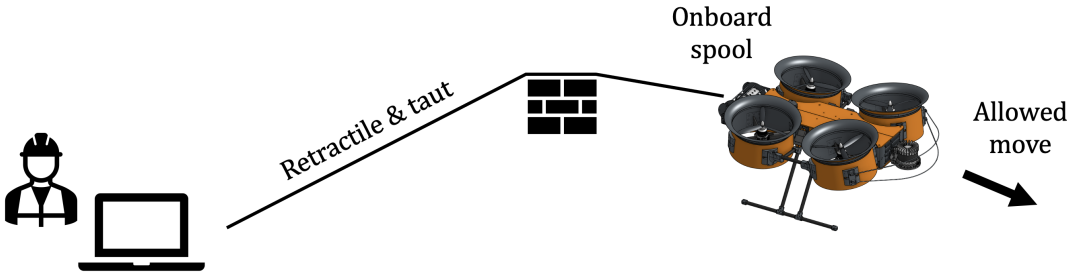


Figure 2.33 Schematic representation of a tangle-compatible tethered drone.

Since it directly affects the drone payload at takeoff, the maximum tether length should be kept small. It is assumed that the user chooses beforehand to embark a length of tether that allows to go to the end of the stope by the shortest possible path, i.e., a perfect unwinding of tether would make it possible to explore the whole stope. Studies have been conducted to try to control this tether and avoid entanglements. A technique has been developed in [Martínez-Rozas et al., 2021, Martínez-Rozas et al., 2022] for motion planning of a UAV, connected by a non-tensioned tether to an Unmanned Ground Vehicle (UGV). The technique is based on a multi-objective optimization with, among others, the trajectory and the tether length (which is dynamically controlled) as parameters. It allows to avoid collisions of the tether with obstacles, as displayed in Fig, 2.34. The disadvantage of this technique is that it considers a collision-free tether. In the case of stope exploration, the drone sometimes has to explore spaces lower in altitude than the takeoff point. This results in tether contact points being necessary to explore the environment. The catenary model of a 3D tether with contact points becomes almost impossible to compute and no work has succeeded, to our knowledge, to model this in real time.

Research has also been done to solve a 2D TSP for given waypoints in a known space with the constraint of a non-entangling path. It has been integrated into an underwater drone as shown in Fig 2.35a. The path between the given waypoints is optimized to avoid tether entanglements with buoys whose positions are known. The objective function is

$$C(T) = k(E_T + 1) * L_T,$$

in which C is the cost, E_T is the number of entanglements and L_T is the length of the tether. It is based on the observation of path homotopy classes (defined earlier in Fig. 2.19) thanks to their identification by the H-signature [McCammon and Hollinger, 2017]. The concept of H-signature is defined in [Bhattacharya et al., 2012]. Fig 2.35b displays a

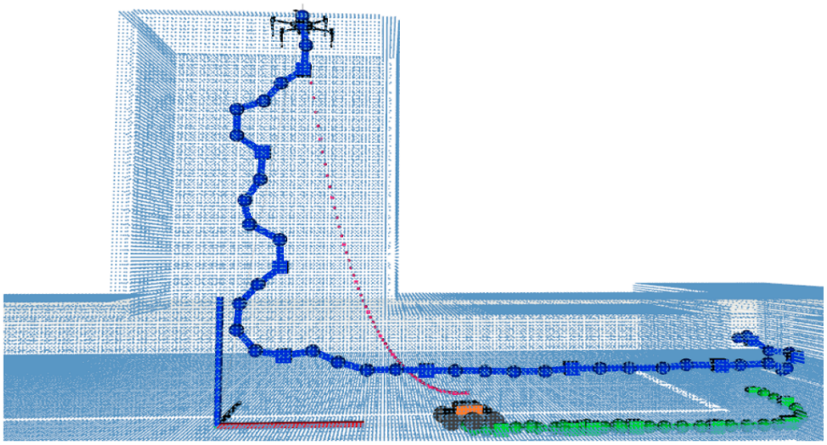
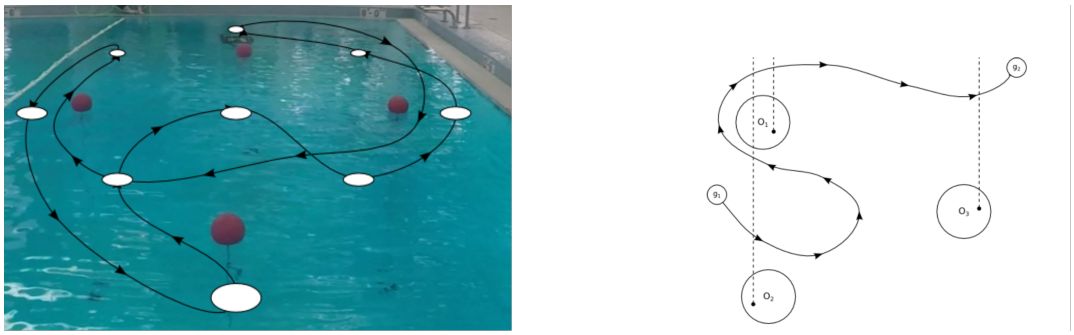


Figure 2.34 Example of the method from [Martínez-Rozas et al., 2022] applied to a marsupial robotic system. The tether is in red.

path that would have "$O_2, O_2^{-1}, O_2, O_1, O_3$" = "$O_2, O_1, O_3$" as H-signature. This indicates that the path crosses the vertical above o_2 to the right, then the same vertical to the left, etc. After simplification, the path crosses the vertical above o_2 to the right, then o_1 and finally o_3. The value of E_T is therefore equal to 3, i.e., the number of vertical crossings of the obstacles after simplification.



(a) (b)

Figure 2.35 Non-entangling 2D path between waypoints: (a) resulting path with an underwater drone, (b) concept of H-signature [McCammon and Hollinger, 2017].

This same identification was used in 2D tangle-free exploration with the nearest frontier method in [Shapovalov and Pereira, 2020a, Shapovalov and Pereira, 2020b]. Fig. 2.36 shows how this technique is implemented. The path $\tau_{r,s}$ computed from the robot position p_r to the goal p_g is homotopic to a path resulting from the concatenation of the tether path τ and the path τ_s from the base station p_b to the goal p_g.

This promising method has two major disadvantages when applied to the problem studied here. First, as said before, the nearest frontier selection approach is too greedy and the

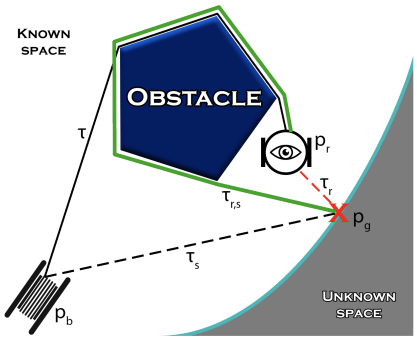
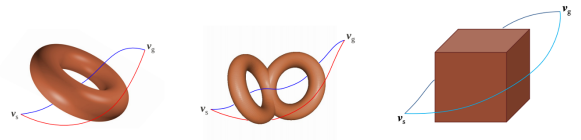


Figure 2.36 Tangle-free exploration to the nearest frontier [Shapovalov and Pereira, 2020b].

global path is therefore not optimal. Second, in 3D, an entanglement may or may not be present for the same homotopy class, and H-signature is therefore irrelevant [Bhattacharya et al., 2010]. Fig. 2.37 provides an example with a solid cube that does not induce homotopy classes. However, one can see that the tether entanglements are not equivalent for the two paths around the cube. Then, it comes down to replacing the 2D homotopy classes information with a proper 3D model of a tether.



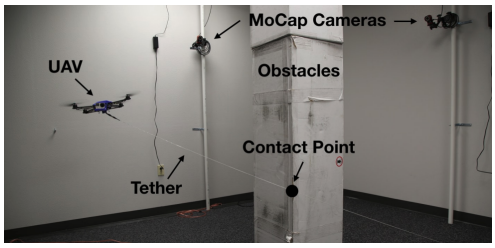
(a) A torus-shaped genus 1 obstacle. (b) A genus 2 obstacle. (c) A solid cube **does not** induce homotopy classes.

Figure 2.37 Homotopy classes in 3D [Bhattacharya et al., 2010].

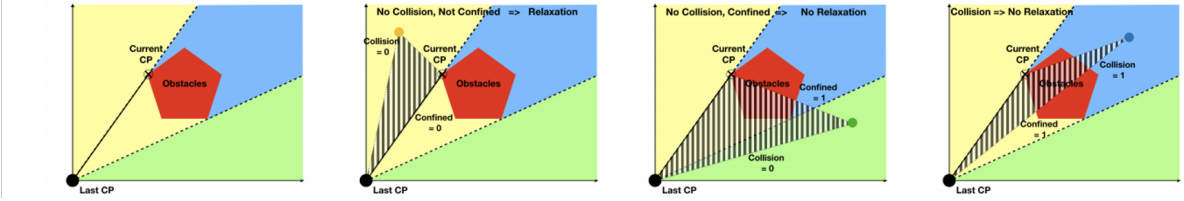
2.2.5 3D tether model

Work has been done to model a tether taut in 3D around an obstacle as pictured in Fig. 2.38a [Xiao et al., 2018]. The model is based on the ray casting information with the known obstacle illustrated in Fig. 2.38b. The path of the drone is traced and the contact points of the tether are recorded in case of collision. Then, if the triangle defined by the last contact point, the current contact point and the drone position is obstacle-free, the tether is relaxed and the current contact point is deleted.

As the method has only been tested with sparse convex obstacles, it has some shortcomings when the drone navigates in a cavity. Fig. 2.39 shows an example of the model in critical cases. If the robot were to follow the path τ_A in Fig. 2.39a, the contact points $CP_{2\to5}$ should be removed almost instantly from one waypoint to the next, but the relaxation



(a)



(b)

Figure 2.38 3D tether model from [Xiao et al., 2018]: (a) experimental setup, (b) model procedure.

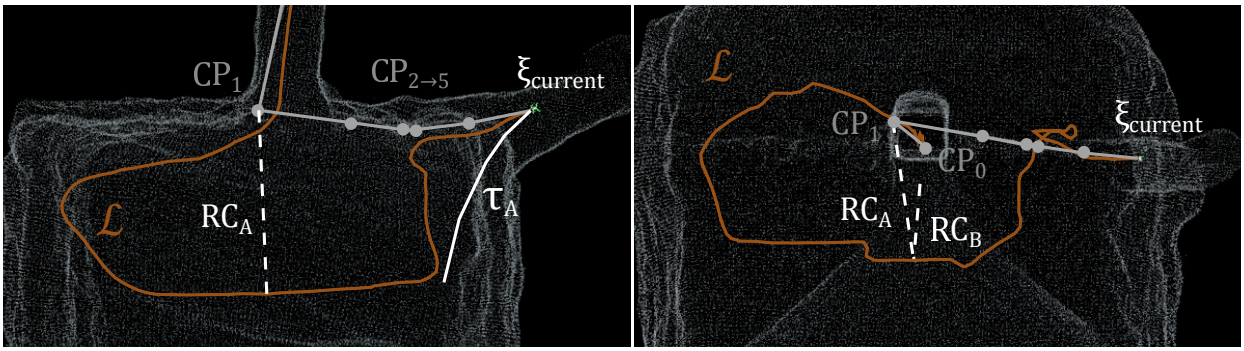
procedure from [Xiao et al., 2018] is implemented for a single point. An improvement to the model for the integration of multi-relaxation is explored in Section 4.4.1.

Also, in 3D, it may happen that the contact points should be adapted when the triangle containing (current contact, last contact, path waypoint) is not collision-free. This is illustrated by the ray casting segment RC_B in Fig. 2.39b which is cut by the wall before reaching CP_0. In reality, the contact point CP_1 should move to the bottom of the central tunnel and then to the right of it, once the robot has reached the position $\xi_{current}$. One approach to solving this problem is to follow the ray casting line RC_A as if they were waypoints in the path history, in order to find the new contact point. This idea is explored in more detail in Section 4.4.1.

2.2.6 Preliminary conclusions

A good exploration algorithm for the problem addressed in this work must minimize the global path length, while remaining under the maximum embedded tether length and running in real time.

Among the exploration techniques, the notion of frontiers is the most interesting to represent the configuration of the space to be explored. On the other hand, a selection technique based on the TSP should be used to optimize the global path.



(a) (b)

Figure 2.39 Illustration of the limits of the model in [Xiao et al., 2018]: (a) top view, (b) front view. The past trajectory $\mathcal{L}$ is in brown and the tether model in gray.

Few exploration techniques consider a tether and those that do are limited to the 2D case and do not focus on optimizing the global path. In 3D, the straight or kinked tether model can be improved for a better representation of reality.

State-of-the-art exploration methods based on a hierarchical architecture are shown to be efficient in decreasing the unhelpful computation time, while keeping a good local navigation accuracy. Moreover, the tether entanglements are strongly influenced by the local behavior but little by the global behavior in the mining stopes.

The exploration problem could thus be broken down into the design of a 2-level algorithm whose a) the global level solves a TSP to optimize the global path through the frontiers and back to the base station, and whose b) the local level minimizes the local path and avoids tether entanglements. This idea is developed in Chapter 4.

2.3 Traveling Salesman Problem

As presented in Section 2.2.3, the frontier selection problem for exploration can be reduced to a TSP. By repeating the TSP computation incrementally with the latest map data, the drone decision approaches the optimal exploration path.

The basic TSP is defined as follows: *Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point* [Montiel Ross and Delgadillo, 2015]. This problem, although simply defined, is one of the most studied problems in computational mathematics, and the biggest open problem in complexity theory [Cook, 2011]. It was historically based on the study of Hamiltonian cycles [10] in polyhedra [Biggs, 1981]. The *Clay Mathematics Institute* considers the solving of the TSP, which would settle the P versus NP problem, as one of the 7 Millennium Problems. The TSP is therefore a platform of choice for studying general methods that can be applied to a wide range of discrete optimization problems.

The applications of the TSP and its variants are numerous around us in everyday life including transportation (e.g., truck dispatching for delivery [Dantzig and Ramser, 1959]), logistics (e.g., production line optimization [Delorme et al., 2019]), biology (e.g., DNA sequencing [Caserta and Voß, 2014]), electronics (e.g., PCB manufacturing [Kumar and Luo, 2003]), communication (e.g., fiber optic network installation [Cosares et al., 1995]) and space research (e.g., imaging sequence of celestial objects [Bailey et al., 2000]). A good algorithm depends on the problem concerned by the TSP or one of its variants, the desired accuracy and the allocated computing time.

This section first focuses on the properties of the problem and presents which instance of the TSP best approximates our problem. Then, the most popular exact and approximate techniques to solve it are presented in Section 2.3.1 and Section 2.3.2. A comparative study is provided in Section 2.3.3 for the described methods applied to the problem studied in this work. Finally, the tether unwinding is considered in the global planning in Section 2.3.4.

Graph properties In the context of the frontier selection problem, the graph is complete [11] and undirected [12]. Considering only the distance to be covered (and not the tether length), the TSP considered here is metric. The triangular inequality principle is there-

10. A cycle that visits each vertex of a graph exactly once.
11. Each pair of graph vertices is connected by an edge.
12. The relations between pairs of vertices are symmetric, so that branch costs are equal regardless of the direction of traversal.

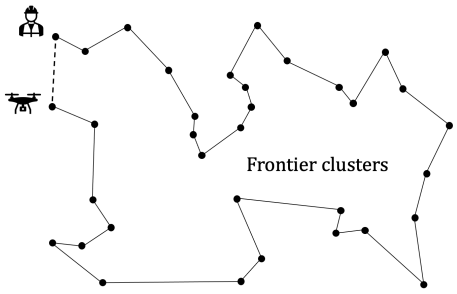


Figure 2.40 Graphical representation of a TSP solution for frontier-based exploration.

fore satisfied, which implies that the solving algorithms do not have to consider paths that pass several times through a node. Finally, the classic TSP considers a path that traverses all cities and returns to the starting point. In this work, the path must have a specific end point, the home point, which is different from the starting point, as shown in Fig. 2.40. This leads to minor variations that have little influence in classical TSP solving algorithms. These are presented below.

Frontier clusters representation One might think that a variant that resembles the problem is the Generalized TSP (GTSP), which consists of finding the shortest path to visit a point for each cluster of points [Hu and Raidl, 2008]. However, this adds a lot of complexity due to the high number of nodes in the problem. Also, a given point in a cluster of frontiers is not necessarily relevant in terms of visibility. An approach of selecting a representative point for each cluster, as detailed in Section 4.4.2, is more appropriate and allows to decrease the complexity of the problem.

Dynamic graph effect The considered problem has similarities with the Canadian Traveler Problem (CTP) in the sense that the graph evolves dynamically during the mission. In the CTP, the graph is partially observable [Bar-Noy and Schieber, 1991]. It is therefore revealed during the exploration but the path traveled is accounted for. It is inspired by the Canadian drivers who may have difficulty navigating a city network with snowfalls that randomly block the roads. However, the graph in this study is even more dynamic since nodes (and thus, branches) are added or removed when frontiers appear or are visited. As mentioned in Section 2.2.3, in the case of frontier-based exploration, the dynamic effect of the graph can be attenuated, since the optimal solution at time t is very close to the optimal solution at time $t+1$. Thus, an effective strategy is to solve the TSP iteratively, with the latest map data. The results of this approach are provided in Section 4.5. Fig. 2.41 shows the graph at a given time during the mission, with branches

in white and frontiers in blue. Each cluster is represented as a single node in the graph. This example illustrates that the map data at time t is very informative of the global path to follow.

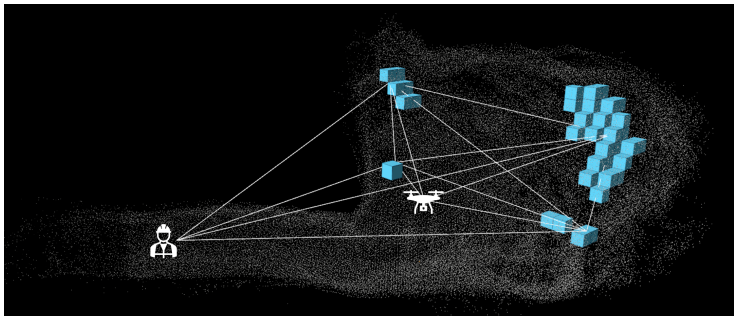


Figure 2.41 Frontier graph at a given time during the exploration.

Solving algorithms The techniques for solving the TSP can be divided into two main categories: exact and approximate algorithms. The problem being NP-hard [13], no exact algorithm in polynomial time is known to date. Exact algorithms often work well for small problems. Approximate algorithms solve the problem in a more reasonable time by providing an approached solution.

2.3.1 Exact solutions

Exact algorithms guarantee the optimality of the solution but can be computationally expensive depending on the problem size. The most common approaches are described in this section.

Brute-force search The naive solution is to try all permutations without repetitions. The time complexity is $\mathcal{O}(n!)$ considering n cities. The solution is quickly too expensive even for a small problem of 10 cities.

Dynamic programming (Held-Karp) A dynamic programming approach was designed independently by [Bellman, 1962] and by [Held and Karp, 1962]. By applying the divide-and-conquer principle, the solution can be computed quickly for a small problem. The idea behind this algorithm is that every subpath in a path of minimum distance is a path of minimum distance between the endpoints of the subpath. In Fig. 2.42, the Hamiltonian path of minimum distance starting from 1 and ending at 6 is

13. A problem is NP-hard (Non-deterministic Polynomial-time Hard) if an algorithm to solve it can be translated into one for solving any other NP problem (Non-deterministic Polynomial-Time), i.e., a problem solvable in polynomial time by a non-deterministic Turing machine. It is therefore "at least as hard as any NP problem".

$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6$. If 1 and 6 are removed, the optimal path starting at 2 and ending at 5 in the set $S = \{2, 3, 4, 5\}$ is $2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5$, which is a subpath of the original optimal path.

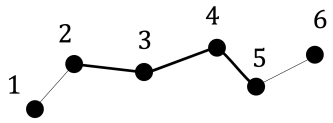


Figure 2.42 Illustration of an optimal subpath in a path.

Repeating the procedure if 5 is removed, the optimal subpath between 2 and 4 in the set $S = \{2, 3, 4\}$ has only one solution. The problem thus becomes trivial by dividing the initial problem into sub-problems. Fig. 2.43 illustrates how the set of size N=5 (first row) is divided into 5 sets of size N=4 (second row). The gray point represents the removed vertex.

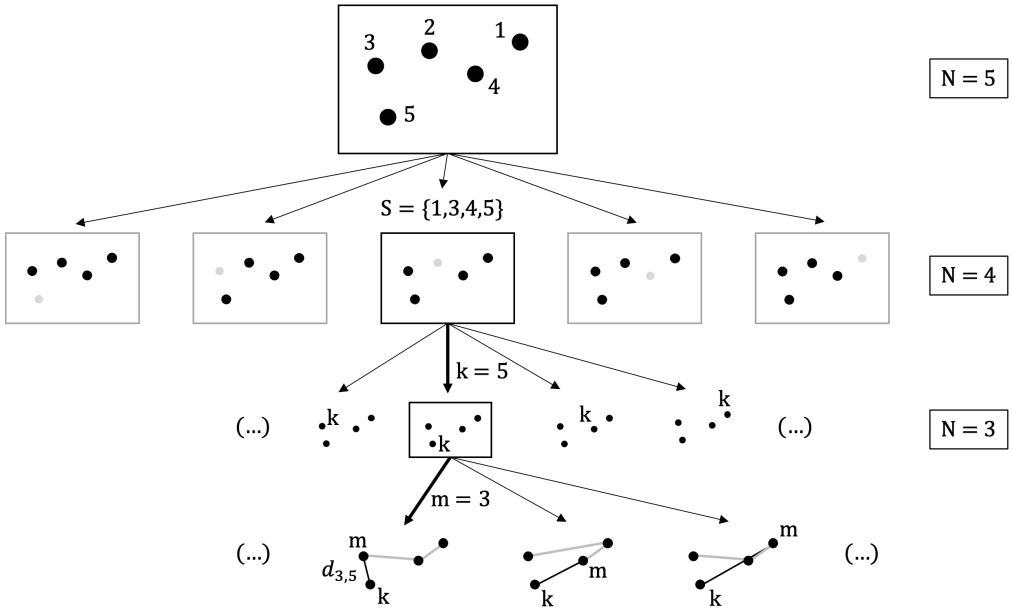


Figure 2.43 Set subdivision tree in Held-Karp.

Held-Karp (HK) is formally presented in Algorithm 1. With the starting city [14] ($k = 1$) previously defined, the first two lines computes the cost g of the subset $S = \{k\}$ of size 1, for a path ending at k. This is equal to the branch cost, previously computed in an adjacency matrix. This matrix is filled with either the Euclidean distance, or the path cost computed with a path planner, hence the importance of quickly computing the path requests, as mentioned earlier. Lines 3 and 4 iterate over all subsets S of increasing size.

14. The starting city is excluded in the size computation.

Line 5 considers each city k in S to be the last city visited in the path. Line 6 finds the city m in the set S, excluding k, that produces the minimum distance path when visited before k. Finally, line 7 finds the best city k to be last visited in the subset of size $n-1$, before returning to the starting city.

Algorithm 1: Held-Karp

```
1 for k ← 2 to n do
2     g({k}, k) ← d(1, k);
3 for s ← 2 to n − 1 do
4     for all S ⊆ {2, ..., n}, |S| = s do
5         for all k ∈ S do
6             g(S, k) ← min_{m≠k, m∈S}[g(S\{k}, m) + d(m, k)];
7 opt ← argmin_{k≠1}[g({2, 3, ..., n}, k) + d(k, 1)];
8 return opt;
```

This considers a Hamiltonian cycle, but a slight modification must be made for the problem considered in this work. As mentioned earlier, the starting and ending city are not equivalent since the problem is recomputed during the exploration mission. The problem then becomes to find a minimum distance Hamiltonian path, and the optimal solution is different from the Hamiltonian cycle for the same problem, as shown in Fig. 2.44. HK can be easily adapted to this problem by removing the term $d(k, 1)$ from line 7.

(a) (b)

Figure 2.44 Adaptation of HK to robotic exploration: (a) optimal tour, (b) optimal exploration path.

For lines 3 and 4, the trick for enumerating all subsets is to represent them by a bitmask of length n. Thus, there are 2^n subsets, among which the procedure iterates over k and over m. The time complexity of HK is thus $\mathcal{O}(n^2 2^n)$.

Branch-and-Bound Also based on the divide-and-conquer principle, the Branch-and-Bound method (BnB) allows to avoid the evaluation of all possible solutions. First, an initial guess is computed with another technique (e.g., greedy heuristic). Then, the solutions are enumerated in the form of a tree, as illustrated in Fig. 2.45.

The technique alternates between branching, i.e., enumerating the remaining solutions in the visiting order, and bounding, i.e., computing an expected lower bound for a given

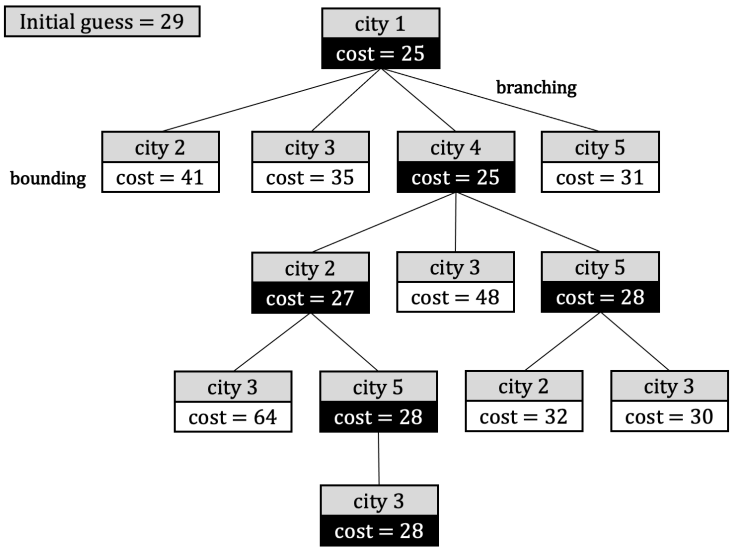


Figure 2.45 Branch-and-bound procedure.

branch. Then, if the bound is greater than the current solution, the branch is dropped. BnB is similar to a BFS in the first instance, and only go deeper into the tree for promising solutions, unlike a DFS. Estimating a lower bound is crucial in BnB to eliminate as many solutions as possible. The lower bound must be as exact (large) as possible. The cost of a tour $A \Rightarrow B \Rightarrow C \Rightarrow D$ is equal to half the sum of the 2 adjacent tour branches (in bold for A in Fig. 2.46) for each node. In mathematical form,

$$c(T) = \frac{1}{2}\left[(\overline{\mathrm{DA}} + \overline{\mathrm{AB}}) + (\overline{\mathrm{AB}} + \overline{\mathrm{BC}}) + (\overline{\mathrm{BC}} + \overline{\mathrm{CD}}) + (\overline{\mathrm{CD}} + \overline{\mathrm{DA}})\right].$$

The lower bound computation is based on this unconventional way of computing a tour cost.

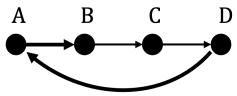


Figure 2.46 Unconventional way of computing a tour cost.

In the graph displayed in Fig. 2.47, a tour will never be shorter than half the sum of the minimum 2 branches adjacent to each city. For example, the 2 minimum branches adjacent to B, in bold, have a cost of 3 and 6. The least expensive way to get to and from B is to traverse these branches.

The time complexity of BnB is lower than $\mathcal{O}(n!)$ as the worst case (i.e., no branch pruning) is equivalent to the brute-force method. The performance of this method depends on the first solution and the evaluation order of the cities.

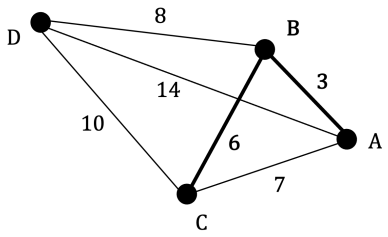


Figure 2.47 Lower bound computation.

Branch-and-cut The branch-and-cut method is a combination of BnB and the cutting-plane method [Gilmore and Gomory, 1961]. This latter, used in Linear Programming (LP), allows to relax the problem by temporarily violating the TSP cycle constraint. This is the method used in the Concorde TSP Solver [Applegate et al., 2006] that holds the current record for the largest solved problem (85,900 cities in 136+ CPU years) with an exact algorithm. More details about the algorithm and the proof of optimality are provided in appendix B.

Time complexity When applied to the problem of this study, Fig. 2.48 shows the running time of BnB and HK depending on the number of frontier clusters (or cities). As expected, BnB has a more variable performance depending on the initial solution and the evaluation order. Both solutions are not viable for running in real-time for a problem larger than 15 cities. To date, no one has determined whether an exact algorithm that runs in $\mathcal{O}(1.9999^n)$ exists [Woeginger, 2001].

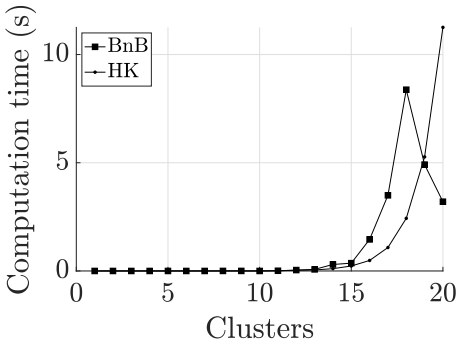


Figure 2.48 Exact algorithms performance for the exploration TSP.

2.3.2 Approximate solutions

For large problems, different approximation and heuristic techniques have been designed to provide a solution more quickly by trading optimality for speed. The most popular ones are described below.

Greedy heuristic The naive (and fastest) approach is the Nearest Neighbor (NN) algorithm. The traveler always chooses the nearest non-visited city and the time complexity is $\mathcal{O}(n^2)$. For a given problem, the solution depends on the starting point. On average, it produces a path 25% longer than optimum [Johnson and McGeoch, 2018]. In some special arrangements, it generates the worst tour [Gutin et al., 2002]. This problem often occurs for the last branches of the tour that were not anticipated.

Christofides algorithm Approximation algorithms, unlike heuristics, promise a worst-case performance and are part of a more formal framework. The most popular approximation algorithm promises an approximation factor of 1.5 [Christofides, 1976]. The general outline of the algorithm and the proof of the approximation ratio are given in appendix C. This literature review does not detail further how this algorithm works since some heuristics do not guarantee an approximation ratio but have much better performance than Christofides [Genova and Williamson, 2017].

2-opt move Once a first tour is found with NN or Christofides, a heuristic can be used to produce a better tour. Fig. 2.49 illustrates the 2-opt move which is a local search that consists in exchanging edges that are part of the tour with edges that are not part of it, in order to create a new legal tour. First, t_1 is selected and a neighbor t_2 is deduced. Then t_3 is selected and a t_4 neighbor is deduced. Finally, t_1 is reconnected to the vertex among t_3 and t_4 which closes the tour (t_4 in the figure), and the same goes for t_2. The traversal order of the half tour is also reversed. This operation is repeated for all edges as long as the new tour is improved.

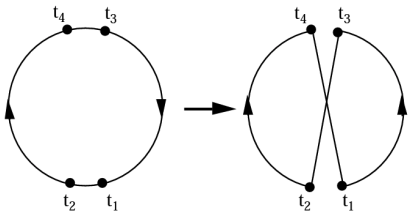


Figure 2.49 2-opt move [Helsgaun, 2000].

3-opt move Fig. 2.50 illustrates the 3-opt move which is a little slower than 2-opt move but produces a better quality solution. The process is similar but a more generic way than choosing the vertices denoted t_i is to note the edges x_i that are removed and the edges y_i that are added.

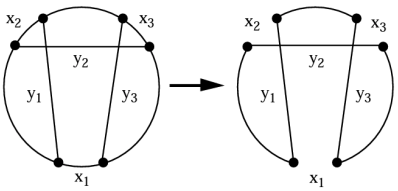


Figure 2.50 3-opt move [Helsgaun, 2000].

Local minima The last two heuristics encounter local optimum problems. For example, a 2-opt move will produce a tour that is only 2-optimal. Fig. 2.51 shows a double bridge move (or non-sequential 4-opt move) that is impossible to achieve with a sequential selection of 2-opt moves or 3-opt moves.

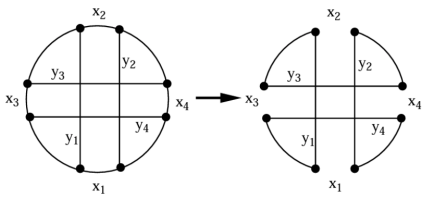


Figure 2.51 Double bridge move [Helsgaun, 2000].

Fig. 2.52 depicts a sequential 4-opt moves performed with back-to-back 2-opt moves.

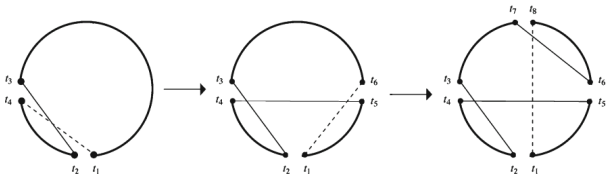


Figure 2.52 Sequential 4-opt move [Helsgaun, 2000].

Graph partitioning The Lin-Kernighan (LK), or k-opt move heuristic is one of the best heuristics for solving the TSP [Lin and Kernighan, 1973]. The first idea of the heuristic was to solve the graph partitioning problem, i.e., in the graph V, drawn in Fig. 2.53, find a partition V_1 and a partition V_2 that have the same number of vertices but are minimally connected by edges. Lin-Kernighan implemented a local search that swaps the subsets X_1 and X_2, leaving only one edge crossing the partition instead of 4 in this example [Kernighan and Lin, 1970].

Lin-Kernighan (LK) The main idea behind LK, applied to TSP is an adaptive k-opt move, in which k is variable depending on the calculated distance gain. Appendix D provides a basic pseudo-code of the procedure because its implementation is relatively

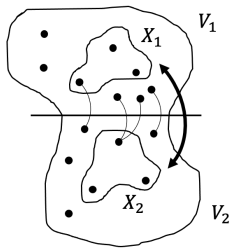


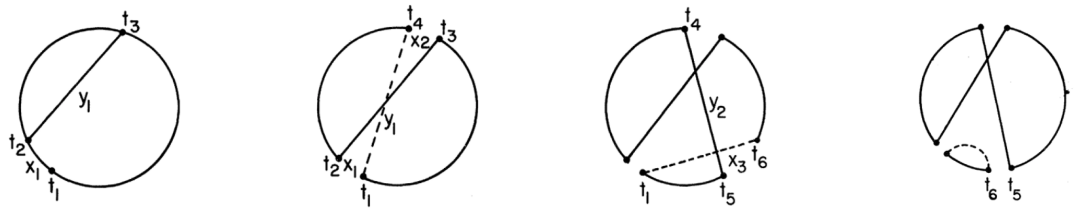
Figure 2.53 Graph partitioning.

extensive. In fact, the optimal implementation in modern code languages uses recursive functions. More info can be found in [Helsgaun, 2000].

Basically, the algorithm starts in the same way as a 2-opt move, i.e., t_1 is chosen randomly, a neighbor t_2 is considered and t_3 is chosen randomly, as shown in Fig. 2.54a. Then, an expected gain is defined by

$$G_0 = 0,$$
$$G_i = G_{i-1} + x_i - y_i.$$

The lengths x_i represent the edges that belong to the tour T and will be removed while the lengths y_i represent the edges that will be added to T. If the gain is negative, the algorithm evaluates an untried alternative for y_1 and x_1. If the gain is positive, the operation continues by choosing a t_4 (i.e., t_{2i}) neighboring t_3 (i.e., t_{2i-1}) and checks if these choices lead to a legal tour T', as illustrated in Fig. 2.54b. If T' is shorter than T, the new tour is kept and the algorithm starts again with a new choice of t_1. Otherwise, the procedure continues for y_i and x_i with an increasing i as illustrated in Fig. 2.54c. Finally, the untried alternatives for x_1, y_1, x_2, y_2 and t_1 are evaluated. Certain conditions, detailed in [Lin and Kernighan, 1973], are necessary to ensure that this sequential procedure produces a tour. A counterexample is provided in Fig. 2.54d.



(a) (b) (c) (d)

Figure 2.54 k-opt move [Lin and Kernighan, 1973].

This heuristic can fall into a local minimum because it only allows to perform sequential exchanges and a path produced by a double-bridge move, as shown in Fig. 2.51, can never be obtained for example.

Chained Lin-Kernighan In the continuity of LK, the Chained Lin-Kernighan (CLK) algorithm has been implemented to overcome the local minimum problem that can occur with LK for large problems [Applegate et al., 2003]. The idea behind CLK is to perform a double-bridge kick, as displayed in Fig. 2.51, when LK does not improve the solution anymore, and to restart LK afterwards.

Randomized improvement The heuristics presented above have been shown to be very effective but they sometimes fall into local minima. To get out of these, various metaheuristics can be used. Since the TSP is a benchmark for optimization algorithms, many combinatorial optimization techniques have been applied to the problem including genetic algorithms [Freisleben and Merz, 1996], ant colony optimization [Dorigo and Gambardella, 1997], tabu search [Khan and Asadujjaman, 2016], and simulated annealing [Pepper et al., 2002]. In this literature review, the choice is made to develop the simulated annealing method because it is relatively efficient compared to other metaheuristics [Bi et al., 2021, Vaishnav and Patil, 2018, Alhamdy et al., 2012].

Simulated annealing A metaheuristic often used in global optimization is Simulated Annealing (SA) [Kirkpatrick et al., 1983]. This technique takes its name from a metallurgical process that involves controlled heating and cooling of a material to increase its crystal size and reduce its defects. Starting from an initial solution, a heuristic (e.g., the 2-opt move) is used to generate an alternative tour. Less good solutions are temporarily accepted to escape local minima. The acceptance depends on a probability

$$p = e^{\frac{\Delta}{T}},$$

where Δ is the cost difference between the two tours, and T is the temperature. A random number r is given between 0 and 1, and the new solution is accepted if it is better, or if it is worse but $p > r$. The temperature T is initialized (e.g., $T_0 = 5000$) and a cooling schedule is provided, following a geometric law

$$T_i = \alpha\, T_{i-1},$$

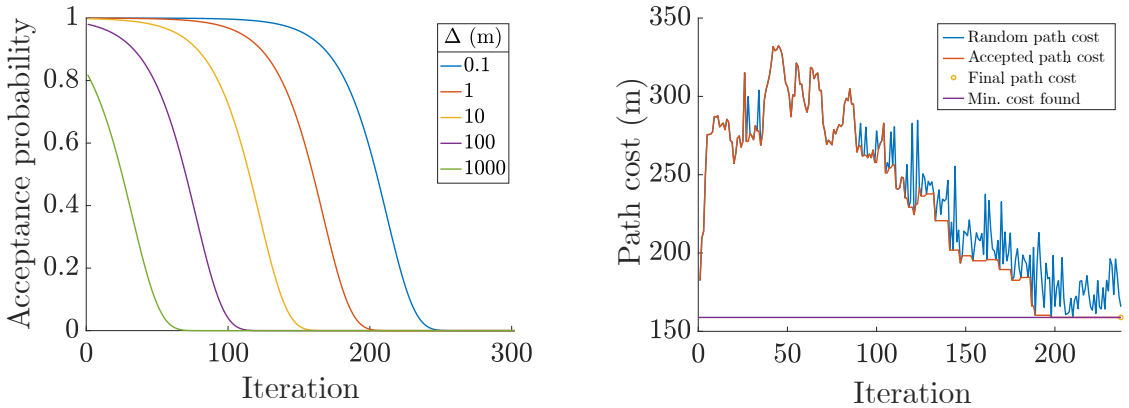
where α is the cooling rate (e.g., $\alpha = 0.95$) [15].

15. A better convergence was observed with a cooling schedule that follows a Lundy decrease law [Lundy and Mees, 1986], i.e.,

$$T_i = T_{i-1}/(1 + \beta\, T_{i-1}),$$

where β is a small constant related to the cooling rate.

The algorithm finishes when the solution has converged or when the temperature reaches a final value, e.g., $T_f = 0.001$. The lower the cooling rate, the better the convergence, at the cost of computing time. Fig. 2.55a illustrates the acceptance probability depending on the iteration and path cost difference Δ. Fig. 2.55b shows the resulting convergence for a TSP of 15 frontier clusters. One can observe that the algorithm selects poorer solutions at the beginning, and ends up selecting only the best among the final random solutions.



(a) (b)

Figure 2.55 Convergence of SA.

2.3.3 Comparative study

Fig. 2.56 presents the results of the most promising algorithms for exploration TSP. These methods were tested several times during various simulated exploration missions, with the number of frontier clusters ranging from 4 to 20. The raw points are approximated by representative curves [16]. HK is used as a ground truth to determine the optimal tour cost. Fig. 2.56a shows the distance to this optimal tour, and Fig. 2.56b shows the computation time. SA was implemented with a 2-opt move heuristic. The results indicate that LK and CLK provide the most promising results, with a maximum approximation factor of 2%, which is negligible for the problem considered. For a number of cities lower than 13, HK is nevertheless faster in computation time. The final solution chosen to solve the TSP is therefore HK when $n < 13$ and LK otherwise.

Fig. 2.57 provides examples of solutions with this method during a simulated exploration mission. The highlighted frontier cluster designates the next cluster to be visited by the drone.

16. The raw points are fitted using the "fit" function in Matlab. The cost is approximated with an "exp2" curve defined by $Y = a\,e^{b\,x} + c\,e^{d\,x}$. The computation time is approximated with a "power2" curve defined by $Y = a\,x^b + c$

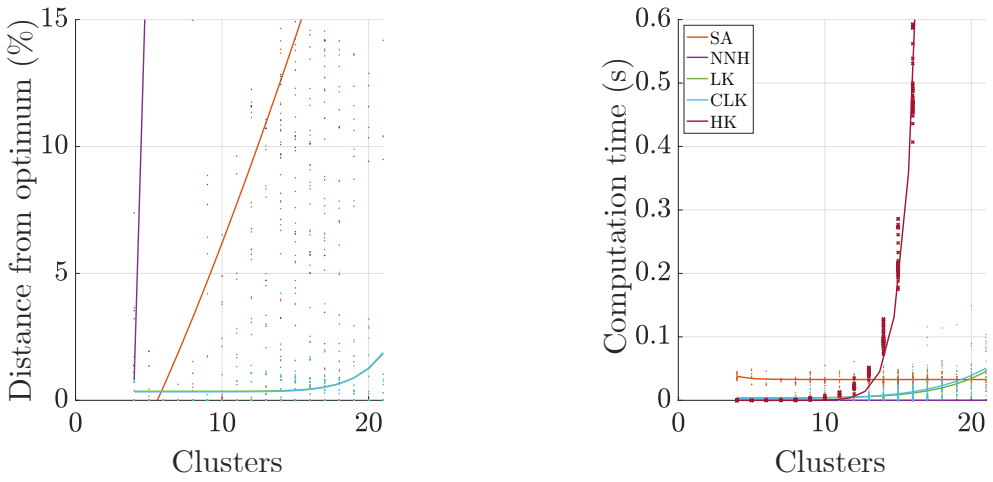


Figure 2.56 Comparison of popular algorithms for the exploration TSP. HK is used as a ground truth to determine the optimal tour cost in (a).

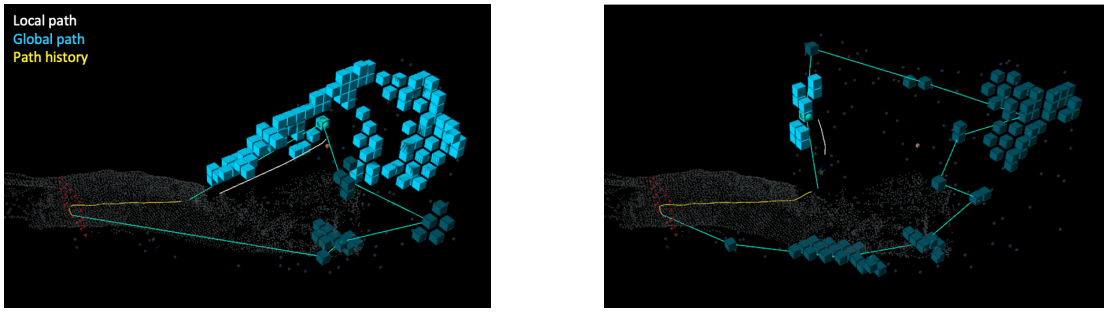


Figure 2.57 Example of TSP solutions for the exploration problem. The optimal visit order evolves in real time as the mission progresses. It provides a global exploration path that is increasingly informed with new map data.

The global path in blue shows the order of visiting the frontiers with straight line segments but as mentioned in Section 2.3.1, the branch costs can be defined by the path cost, calculated with a path planner. In this case, the danger is if some path requests are not successful. This implies that some branch costs are undefined and therefore, the graph is incomplete. If this happens for a lot of branches—i.e., the graph is sparse—one has to make sure that a tour exists. The problem of proving the existence of a Hamiltonian cycle in a sparse graph is NP-hard itself [Garey and Johnson, 1979]. A pragmatic solution is to filter the frontier clusters that cause a lot of failed path requests, since they are probably non-accessible clusters.

2.3.4 Tether consideration

By taking into account the tether in the global path optimization, the definition of the TSP changes from the classical problem. Since the tether configuration at a given position

depends on the path and changes for each solution, the only viable solution among those presented is randomized improvement (e.g., SA). Most of the other methods take advantage of problem specific tricks that are not applicable with the tether. Moreover, for a 2-opt move for example, the tether configuration must be recalculated for all positions following the position of t_1, and this can be computationally heavy. This technique is tested in Chapter 4 but fortunately, the results show that the global path does not influence the tether entanglements and the classical TSP is therefore representative of the problem.

CHAPTER 3

RRT-Rope

Preamble

Title: RRT-Rope: A deterministic shortening approach for fast near-optimal path planning in large-scale uncluttered 3D environments.

French title: RRT-Rope : une approche déterministe de raccourcissement pour la planification rapide de chemins quasi-optimaux dans des environnements 3D non encombrés à grande échelle.

Authors and affiliation:

L. Petit: Ph.D. candidate, Université de Sherbrooke, Faculty of Engineering, Department of Mechanical Engineering. Createk Design Lab.

A. Lussier Desbiens: Professor, Université de Sherbrooke, Faculty of Engineering, Department of Mechanical Engineering, Createk Design Lab.

Date of acceptance: 24 July 2021

Acceptance status: Final published version

Journal: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)

Awards: Best student paper and top 5 best paper

Reference: [Petit and Lussier Desbiens, 2021]

Contribution to the document: This paper contributes to the thesis by developing a path planning algorithm capable of executing quickly and finding a quasi-optimal path in large-scale uncluttered 3D environments by leveraging the knowledge of the problem. This solution is better positioned than the state of the art techniques in terms of execution speed, results accuracy and results precision. This algorithm makes it possible to minimally meet the mission objectives 1 (RTH) and 3 (navigation to a waypoint chosen by the pilot) defined in Section 1.4. In addition, the algorithm will be used for path queries performed in the global and local planner of the exploration algorithm presented in Chapter 4.

Note 1: Concerning the initial requirement of real-time computing introduced in section 1.3, the maximum path request time should be about 0.03 seconds. The reader may notice that this requirement is not respected in the results presented in Fig. 3.7. Following the publication of this paper, internal software optimizations were made to speed up the process. With these changes, the time for a path request is about 0.01 seconds. This is

mainly due to the data logging that was performed for the comparison of the algorithms and imposed a low frequency execution. Moreover, the tree that was kept in memory turns out to be useless since the path always converges to a quasi-optimal path in mining stopes. The shortcut is therefore calculated directly on the output path instead of the tree. These improvements do not influence the orders of magnitude of the comparative results of the algorithms presented in Fig. 3.7. Therefore, the article has not been modified in its present form compared to the published version.

Note 2: Following corrections requested by the members of the jury, the content of this article differs slightly from the one that was accepted.

Abstract: Many path planning algorithms have been introduced so far, but most are costly, in path cost and in processing time, in large-scale uncluttered 3D environments such as underground mining stopes explored by an unmanned aerial vehicle (UAV). Rapidly-exploring Random Tree (RRT) algorithms are popular because of their probabilistic completeness and rapidity in finding a feasible path in single-query problems. Many of the algorithms (e.g. Informed RRT*, RRT#) developed to improve RRT need considerable time to converge in large environments. Shortcutting an RRT is an old idea that has been proven to outperform RRT variants. This paper introduces a new method, RRT-Rope, that aims at finding a near-optimal solution in a drastically shorter amount of time. The proposed approach benefits from fast computation of a feasible path with an altered version of RRT-connect, and post-processes it quickly with a deterministic shortcutting technique, taking advantage of intermediate nodes added to each branch of the tree. This paper presents simulations and statistics carried out to show the efficiency of RRT-Rope, which gives better results in terms of path cost and computation time than other popular RRT variations and shortening techniques in all our simulation environments, and is up to 70% faster than the next best algorithm in a representative stope.

Résumé français: De nombreux algorithmes de planification de trajectoire ont été introduits jusqu'à présent, mais la plupart sont coûteux, en longueur de chemin et en temps de calcul, dans des environnements 3D non encombrés à grande échelle tels que les chantiers miniers souterrains explorés par un drone. Les algorithmes d'arbres aléatoires à exploration rapide (RRT) sont populaires en raison de leur complétude probabiliste et de leur rapidité à trouver un chemin réalisable dans les problèmes à requête unique. De nombreux algorithmes (par exemple, Informed RRT*, RRT#) développés pour améliorer les RRT ont besoin d'un temps considérable pour converger dans des environnements de grande taille. Raccourcir un RRT est une vieille idée qui a prouvé qu'elle était plus performante que les variantes de RRT. Cet article présente une nouvelle méthode, RRT-Rope, qui vise à trouver une solution quasi-optimale en un temps radicalement plus court. L'approche proposée bénéficie du calcul rapide d'un chemin réalisable avec une version modifiée de RRT-connect, et le post-traite rapidement avec une technique déterministe de raccourcissement, en tirant parti des nœuds intermédiaires ajoutés à chaque branche de l'arbre. Cet article présente des simulations et des statistiques réalisées pour montrer l'efficacité de RRT-Rope, qui donne de meilleurs résultats en termes de coût du chemin et de temps de calcul que d'autres variations populaires de RRT et techniques de raccourcissement dans tous nos environnements de simulation, et est jusqu'à 70% plus rapide que le deuxième meilleur algorithme dans un chantier minier représentatif.

3.1 INTRODUCTION

The challenges of 3D path planning have been addressed by many [Yang et al., 2014], with Rapidly-exploring Random Tree (RRT) algorithms being commonly used. However, although RRTs require low computation time to find a first collision-free trajectory, the time required to find an optimal (Euclidean shortest) path can be very large, especially in large-scale uncluttered 3D environments such as underground mining stopes. The problem of navigating through stopes can be considered holonomic for a mapping unmanned aerial vehicle (UAV), as described in section 3.2.1. Node based algorithms like A* [Hart et al., 1968] are frequently used for 3D holonomic path planning but the computational complexity makes them slow in large environments for small discretization of the 3D space, as shown in section 3.5.1, while large discretization makes them incomplete or non-optimal.

Traditional RRTs [LaValle and Kuffner, 2000] are based on generating random points in the reachable space. A connection is then established from the nearest node in the tree x_{near} to a point x_{new}, which is located on the line between x_{near} and a point of randomly generated coordinates x_{rand}, at an infinitesimal step ϵ from x_{near}. One of the main reasons why RRTs are so widely used is that they are biased towards search into the largest Voronoi regions in a graph of the configuration space. In holonomic problems, the infinitesimal step size (ϵ) can be omitted, and x_{rand} used as x_{new}, due to the high probability that the generated random point is reachable from the nearest node [Lindemann and LaValle, 2001]. RRT algorithms are popular because of their probabilistic completeness and quick path computation in single-query problems, but they fail to converge to optimal solutions because of existing graph bias.

RRT* is an evolution of RRT that ensures asymptotic optimality [Karaman and Frazzoli, 2011]. It adds recursive rewiring to RRT by connecting new nodes as parent nodes of a neighbour node if the latter's cost is improved. Informed RRT* [Gammell et al., 2014] improves the convergence rate and solution quality by generating new random points in an ellipsoidal subset of the space. Other ideas for online path optimization, including RRT# [Arslan and Tsiotras, 2013], C-FOREST [Otte and Correll, 2013], searching around beacons [Kanehara et al., 2007], and graph pruning [Karaman et al., 2011, Gammell et al., 2018] have been presented. First, it is shown in section 3.5.1 that the final path cost of RRT-Rope does not depend on initial path quality and online optimization is therefore obsolete. Second, if the goal position is challenging to reach, all these algorithms need large computation time to find a first path. To that end, RRT-connect [Kuffner and LaValle, 2000] was developed to improve path-finding time by growing two trees simultaneously

toward each other: T_A, a tree from x_{start}; and T_B, a tree from x_{goal}. This algorithm is efficient for fast single-query path finding, but not optimal.

Post-processing techniques have been introduced to overcome this problem. Shortcutting an RRT is an old idea that has been proven to outperform optimal RRT variants [Luo and Hauser, 2014]. Node pruning, based on triangular inequality, has been used on A* trajectories [Kanehara et al., 2007], but A* inherent computational complexity is an apparent disadvantage in high dimensions. Node pruning is also used with RRT* [Islam et al., 2012]. However, the resulting path is not optimal because the node resolution is reduced through node pruning.

Partial-shortcut [Geraerts and Overmars, 2004] is one of the most popular shortening methods. It selects 2 arbitrary points in a path and moves the path segment between them on a straight line (or linear interpolation for multiple DOF systems) if there is no collision. Since points are moved instead of pruned, the path is closer to optimality. However, computation time is not optimal due to some irrelevant shortcuts performed because of the non-deterministic point selection (e.g. for an already straight line path or for a to-be pruned portion of path).

An iterative technique based on bisection was developed in [Berchtold and Glavina, 1994] but it could not find a better solution located far away and will only return convex solutions.

Elastic strips [Brock and Khatib, 2002] is another iterative technique that operates by locally displacing a point according to a virtual attractive force towards its two neighbours and a repulsive force coming from obstacles. Since the algorithm is designed for online control, the convergence is slow when used as a shortening method, even with inertial gradient descent.

Finding a feasible path is thus generally very efficient with RRT-like techniques like RRT-connect, but optimization remains lengthy in high-dimensional problems. In large configuration spaces, most shortening methods require many iterations. Moreover, the algorithm typically ends when the computation time exceeds a value fixed by the user, or when the measured convergence rate is small enough. This makes for variable and unpredictable performance.

A new method, *RRT-Rope*, is introduced in this paper to rapidly optimize the first path found with an RRT-connect variant and a deterministic node selection.

This paper is structured as follows. Section 2 establishes the notation and formal definition of the problem. Section 3 details the operation of the RRT-Rope algorithm. In section 4, the algorithm's complexity and optimality are analyzed. Section 5 presents the results of our simulations with a mapping UAV in large stopes. A comparison with other methods is also presented—with a focus on convergence, performance relative to the other shortening methods cited in this section, and step-size sensitivity—to highlight the RRT-Rope algorithm's efficiency in stopes. Finally, this paper concludes with potential extensions to the RRT-Rope algorithm.

3.2 PRELIMINARY MATERIAL

3.2.1 Assumptions

Since the UAV is used for mine mapping, the problem of path planning in mine stopes relies on the following assumptions. For a UAV, the Euclidean space is $SE(3) = \mathbb{R}^3 \times SO(3)$ for a 6-parameter configuration: three positions (x, y, z) and three angles (ϕ, θ, ψ). Since a mapping drone would likely travel at low velocities, the roll (ϕ) and pitch (θ) angles are close to zero. Furthermore, at slow speeds, the yaw angle (ψ) can be controlled independently to, e.g., always let the sensors point forward. Finally, the obstacles in the environment are enlarged offline by the drone radius and a safety factor. This allows us to consider the drone as a single point. For these reasons, the search space can be reduced to $\mathbb{R}^3$ for a point-like problem and dynamic constraints can be ignored. Therefore, RRT can be used without infinitesimal step size (ϵ) as mentioned in [Lindemann and LaValle, 2001].

3.2.2 Notation

Following the notation introduced in [LaValle, 2006], we can define the world $\mathcal{W}$, a semi-algebraic obstacle region $\mathcal{O} \in \mathcal{W}$, a semi-algebraic UAV $\mathcal{A}$ defined in $\mathcal{W}$ as a point, and q the configuration of $\mathcal{A}$. In the case of a drone exploring stopes, $\mathcal{W} = \mathbb{R}^3$ and $q = (x, y, z)$. The configuration space C can be decomposed into the free space C_{free} and the obstacle region C_{obs}. Practically, C_{free} and C_{obs} are obtained with the voxel occupancy probability in an OctoMap [Hornung et al., 2013] generated from $LiDAR$ measurements enlarged by the drone radius. The obstacle region $C_{obs} \subseteq C$ is defined as $C_{obs} = \{q \in C \mid \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}$. The resulting set of permissible sets is $C_{free} = C \setminus C_{obs}$. Since C_{free} is an open set and C_{obs} is closed, if $\mathcal{A}$ touches $\mathcal{O}$ only by a boundary intersection,

$$\text{int}(\mathcal{A}(q)) \cap \text{int}(\mathcal{O}) = \emptyset \text{ and } \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset, \tag{3.1}$$

where $\text{int}(\mathcal{S})$ denotes the interior of $\mathcal{S}$.

The query pair (q_I, q_G) includes the initial configuration $q_I \in C_{free}$ and the final configuration $q_G \in C_{free}$.

3.2.3 Problem formulation

For feasible path planning, a complete algorithm must compute a continuous path, $\tau : [0,1] \to C_{free}$, such that $\tau(0) = q_I$ and $\tau(1) = q_G$ or correctly report that such a path does not exist. In our case, a path always exists thanks to the stope geometry pictured in Fig. 3.1.

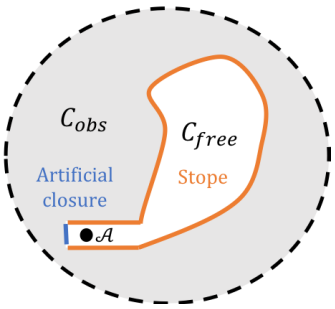


Figure 3.1 Path-planning problem in a mine stope.

The topological space C_{free} is bounded, connected (i.e., it cannot be represented as the union of two or more disjoint non-empty open sets) and path-connected, since for all $q_I, q_G \in C_{free}$, there exists a path τ such that $\tau(0) = q_I$ and $\tau(1) = q_G$. In stopes, there is no column running from one end to the other. Then, C_{free} is simply connected (i.e., any loop can be reduced to a point without encompassing any domain not included in the space). For the definition to ban rocks in suspension, it will be said that the stope surface is connected, and the stope volume is simply connected. Finally, the stope geometry can be considered as mainly wide, from 3 to 30 m wide and 30 to 200 m deep. As [Luna et al., 2013] stated, shortcutting is unlikely to discover a new homotopy. Considering that mine stopes are 3-ball homotopic environments, all paths are homotopic and thereby any path is in the optimal homotopy class.

3.3 RRT ROPE

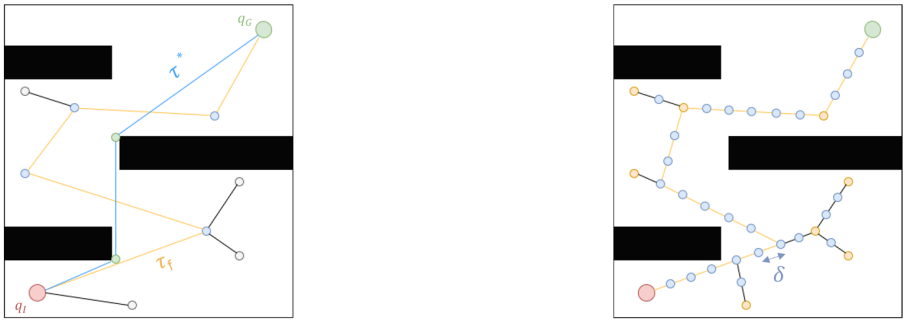
For problems where τ can be a straight line (trivial paths), optimality is met, and the path cost is the Euclidean distance in $\mathbb{R}^3$. In any other case, if τ is optimal, there is at least one $\tau(i) = q_i$ with $i \in [0,1]$, such that (3.1) is verified. The path must then be defined on the closure of C_{free}.

In a simple environment, an RRT without infinitesimal step size (ϵ) could find the yellow path in Fig. 3.2a. Meanwhile, the blue path would be the optimal path for the point-like problem without dynamic constraints. This path is composed of straight lines and parts

of the closure of C_{free}. Since RRT-connect has been found to be fast for single-query path planning, one could optimize the first feasible path found by finding the mapping $\tau_f \rightarrow \tau^*$. RRT-Rope achieves this in a way that is similar to applying tension to a rope.

3.3.1 Path finding

The first step of RRT-Rope is similar to RRT-connect [Kuffner and LaValle, 2000] with the exception that it omits ϵ and inserts intermediate nodes to each branch, during the path search, with a step size δ between them as shown in Fig. 3.2b. Since a non-collision ray casting check has first been performed to wire a branch, it is obvious that all points of a tree branch are in C_{free}. For holonomic path planning in large environments, this is faster than RRT with ϵ (assumed similar to δ) that would require more iterations of random node generation.



(a) (b)

Figure 3.2 Path planning in a simple environment. Fig. (a) shows the first path found by RRT (yellow) and the optimal path (blue) from q_I (red) to q_G (green). Fig. (b) shows the same feasible path with intermediate nodes (blue).

3.3.2 Path optimization

The idea of the rewiring algorithm (depicted in Fig. 3.3) benefits from intermediate nodes to try to get as close as possible to the closure of C_{free}. This is achieved through a method reminiscent of the tightening of a rope. Since the problem is holonomic, the optimal path will be the special case of a Dubins path that does not impose any minimum radius for the robot. In these conditions, the minimal radius of the rope turns is imposed by the obstacles if they are smooth, or is zero otherwise.

For each path node i, taken one by one from q_I to q_G, a collision check is performed with each node j starting from q_G to i. It takes advantage of the width of the environment by checking the farthest nodes first, such as a rope whose curvature would be constrained by obstacles, which ensures to only perform relevant shortcuts, unlike random selection. If the shortcut is collision-free, intermediate nodes will be inserted and the (i, j) pair will

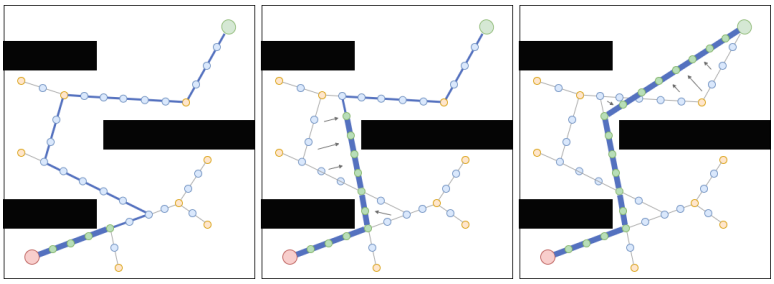


Figure 3.3 Sequence of shortening algorithm, from left to right.

be connected by a straight line. Thus, node resolution is ensured to be constant along the path unlike some other algorithms. For the sake of efficient computation, when the path is already a straight-line between i and j, this will be bypassed to enhance memory management. In both cases, the next collision check will start at $i+1$. This is performed by Algorithm 2.

Algorithm 2: betterPathCost(q_i, q_j)

```
1  if (q_i, q_j) ∈ T_A or (q_i, q_j) ∈ T_B then
2  |   return dist(q_i, q_j) + q_i.cost < q_j.cost;
3  else if q_i ∈ T_A and q_j ∈ T_B then
4  |   return dist(q_i, q_j) + q_i.cost_A + q_j.cost_B < τ_raw.cost;
```

Algorithm 3: RRT-Rope(δ)

```
1  (T_A, T_B) ← RRT-connect(δ, ϵ = false);
2  τ_raw.init(treeToPath(q_I, T_A));
3  τ_raw.add(treeToPath(q_connect, T_B));
4  for i ← 0 to length(τ_raw) do
5  |   τ_opti.add(τ_raw[j]);
6  |   for j ←length(τ_raw) to i + 1 do
7  |   |   if collisionFree(q_i, q_j) then
8  |   |   |   if betterPathCost(q_i, q_j) then
9  |   |   |   |   if τ_raw[i] ∈ T_A and τ_raw[j] ∈ T_A then
10 |   |   |   |   |   T_A.insertIntermediateNodes(δ, q_i, q_j);
11 |   |   |   |   |   τ_raw.init(treeToPath(q_i, T_A);
12 |   |   |   |   |   τ_raw.add(treeToPath(q_connect, T_B));
13 |   |   |   |   else
14 |   |   |   |   |   T_B.insertIntermediateNodes(δ, q_i, q_j);
15 |   |   |   |   |   τ_raw.init(treeToPath(q_i, T_B);
16 |   |   |   |   i ← −1;
17 |   |   |   break;
18 return τ_opti;
```

Algorithm 3 shows the full RRT-Rope procedure. It operates directly on the trees instead of the path, since the nodes could be used for subsequent RRT rewiring (such as Informed

RRT*), as discussed later in section 3.6, but one could use a simplified version on a given path. For the sake of memory management, τ_{raw} only stores indexes of nodes and whether it belongs to T_A or T_B.

There are three possible shortcuts with RRT-connect: inside T_A, inside T_B, or between T_A and T_B. Lines 10-12 handle the first case, whereas lines 14-15 handle the two other cases. If $\tau_{raw}[i] \in T_A$ and $\tau_{raw}[j] \in T_B$, the intermediate nodes of the new branch will be added to T_B and the connection node will change accordingly to become q_i.

Fig. 3.4 shows a simulation performed in a 3D stope. One can see that only three branches (two green in tree A, and one pink in tree B) were needed to find a first path. The light green branches come from RRT-Rope's back-to-back shortcuts.

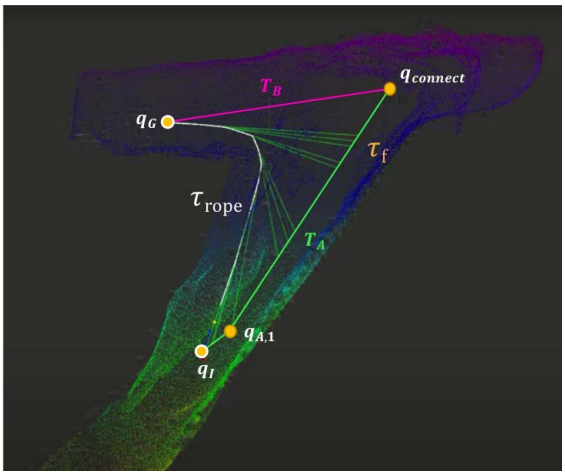


Figure 3.4 Resulting path (white) of the RRT-Rope algorithm in a stope, after optimizing tree A (green) and tree B (pink).

3.4 ANALYSIS

This section presents an analysis of the algorithm. The complexity of adding intermediate nodes and the optimization step are evaluated in the first sub-section, then completeness and optimality are discussed.

3.4.1 Complexity

The algorithm's complexity is composed of two parts: search and optimization.

Path finding

As stated in [Karaman and Frazzoli, 2011] for RRT and RRT*, the space complexity is $\mathcal{O}(n)$ and the time complexity is $\mathcal{O}(n\,log(n))$, expressed as a number of samples n.

Here, n is considerably smaller than usual, since ϵ (i.e., the infinitesimal step between x_{near} and x_{new} in classic RRT) can be ignored for holonomic problems. However, by

adding intermediate nodes, the complexity should rise. In this case, space complexity can be expressed as $\mathcal{O}(n')$ with n', the number of nodes defined in (3.2) as

$$n' = \sum_{i}^{n} ceil(\frac{\text{dist}(q_{p,i}, q_{c,i})}{\delta}) \simeq n\frac{\overline{\text{dist}}(q_p, q_c)}{\delta}, \tag{3.2}$$

where $q_{p,i}$ and $q_{c,i}$ are the parent and child nodes of branch i, respectively, and δ is the step size. If we were to choose $\delta = \epsilon$ with a branch distance equal to ϵ as in classic RRT, $n' = n$ and the space complexity would be the same as RRT. Since the operation of intermediate nodes insertion is bypassed when $\delta = \epsilon$, the time complexity is also unchanged. We can assume that a tree with intermediate nodes on straight line branches is roughly the same as a classic tree with the same infinitesimal step size. The main difference is that it requires fewer random sample generations (none for intermediate nodes) so that n is considerably smaller, as anticipated. However, a bigger number of failed iterations could lead to a rise in n, especially in complex problems, leading to a value of n closer to the usual one. Finally, with RRT-Rope starting with the first path found by RRT instead of an optimized path, n is once again considerably reduced.

Path optimization

Several cases will be used to assess complexity with upper and lower bounds: (a) no shortcut is found; (b) only significantly small shortcuts are found between nodes that are one waypoint apart, resulting in a constant number of waypoints before and after optimization; and (c) a straight-line shortcut exists between q_I and q_G.

The time complexity upper bound is $\mathcal{O}(\frac{m\,(m+1)}{2}) \simeq \mathcal{O}(m^2)$, expressed as a number of path waypoints m:

$$m = ceil(\frac{c(\tau_f)}{\delta}). \tag{3.3}$$

This upper bound considers case (a) and (b). These cases are equivalent, since collision checks must be performed by ray casting at a step size that is similar to δ. The operations of node adding and collision checking are very similar in terms of computation time. So, for either case, $1 + 2 + 3 + ... + m$ operations need to be performed to check each pair (q_i, q_j) and the intermediate nodes between them. The lower bound for time complexity is $\Omega(m_{rope})$ for case (c), with m_{rope} defined as follows:

$$m_{rope} = ceil(\frac{c(\tau_{rope})}{\delta}) \leq m. \tag{3.4}$$

The upper bound for space complexity is $\mathcal{O}(\frac{m\,(m+1)}{2}) \simeq \mathcal{O}(m^2)$ for case (b), and the lower bound is $\Omega(0)$ for case (a).

Since the ranges between lower and upper bounds are wide and the value of m depends on δ, the processing time will be analyzed based on simulation results in section 3.5.3.

3.4.2 Completeness and optimality

Since probabilistic completeness is inherited from the algorithm used for the search operation, it is ensured with RRT-like algorithms. Thus, one can be sure to find a path if it exists. Also, monotonous convergence is ensured with line 8 of Algorithm 3 and with triangular inequality.

Optimality is an extension of the feasible path planning problem that was defined in section 3.2.3. Let Σ be the set of all nontrivial paths. Optimal path planning can be defined similarly to [Karaman and Frazzoli, 2011] and [Gammell et al., 2014], as the search for a path τ^* that minimizes the cost function $c : \Sigma_{C_{free}} \to \mathbb{R}_{\geq 0}$, which assigns a non-negative cost to all nontrivial paths in which each point is on the closure of C_{free}. Optimality is ensured when a feasible path with minimal cost is found. Given C, C_{obs}, q_I and q_G, an optimal algorithm must find a path $\tau^* : [0,1] \to cl(C_{free})$ such that (a) $\tau^*(0) = q_I$ and $\tau^*(1) = q_G$ and (b) $c(\tau^*) = \min_{\tau \in \Sigma_{cl(C_{free})}} c(\tau)$, and report failure if no such path exists.

Asymptotic optimality is not guaranteed by RRT-Rope if used as it is, because of the local minimum that can occur in some specific situation. However, the main goal of the algorithm is to speed up the search for a better path. Also, the extension discussed in section 3.6 could ensure asymptotic optimality at the cost of extra computing time.

3.5 SIMULATION

The approach was integrated in ROS (Robot Operating System) and tested on simulated environments in Gazebo that were reconstructed from real LiDAR scans. The simulated UAV is equipped with a rotating Ouster LiDAR. A simultaneous localization and mapping (SLAM) approach called RTAB-Map [Labbé and Michaud, 2018] is used to provide the OctoMap and localization inputs for the RRT-Rope algorithm. Simulations were performed with an HP Z440 Workstation with 12 Intel Xeon processors running at 3.5 GHz, 24 GB of RAM, with a loop frequency of 1 MHz for path planning iterations. The simulated environments and associated query pairs (q_I, q_G) are pictured in Fig. 3.5.

The goal was to reproduce a classic return-to-home operation after UAV exploration. These three environments have been chosen to best represent the spatial characteristics of

some typical missions. *Env. 2* represents a classic stope environment, with a cylindrical shape and tunnels at the top. The path should be between 20 and 30 m long. *Env. 1* is a bit different, since the take-off point is farther from the stope and the UAV has to move through a long tunnel first. The stope is also different in that it is more like a long inclined flat space and the full path length is between 30 and 40 m. Finally, *Env. 3* is used to establish how well the algorithm works even in an environment that does not meet all the assumptions stated in section 3.2.1, since it is an indoor environment with columns and doors, with path lengths between 40 and 50 m.

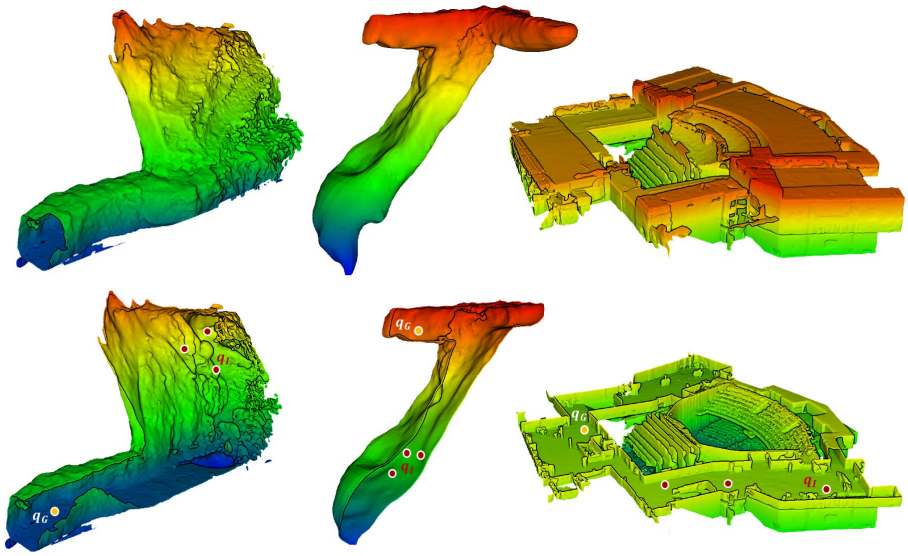
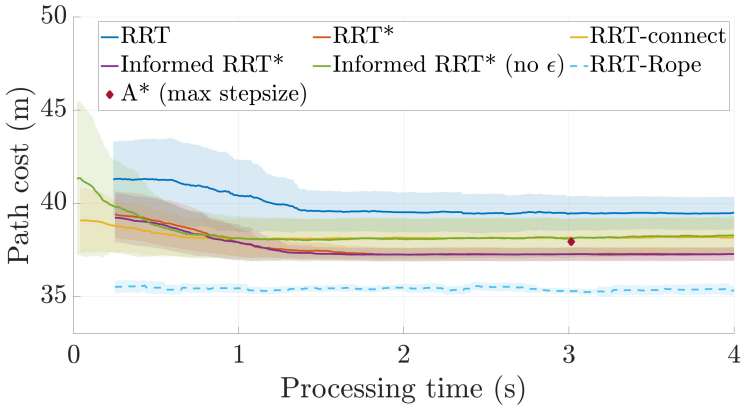


Figure 3.5 External (top row) and cross-section (bottom row) views of the three testing environments: 1 (left), 2 (center), and 3 (right). Note that the points q_I and q_G used in the next simulations are respectively in red and yellow.

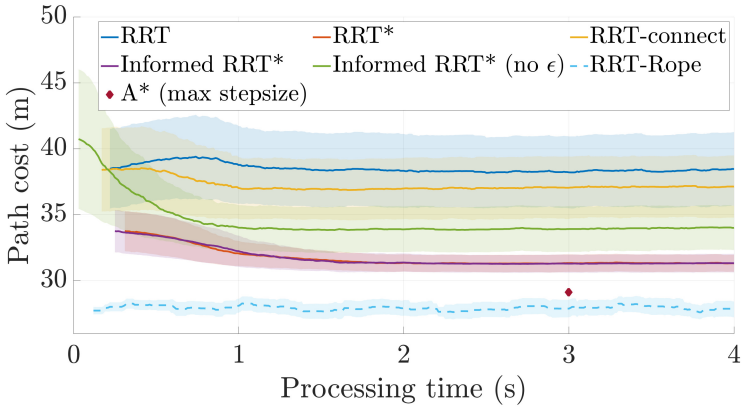
3.5.1 Convergence

First, Fig. 3.6 shows the results from a simulation used to demonstrate the algorithm's fast convergence. These are the results obtained from running RRT and A* algorithms compared to RRT-Rope optimization applied on a path obtained with Informed RRT* with intermediate nodes and initialized at a different time between the moment a path was found and 5 s. This test had two goals: (a) to confirm the performance of using RRT-Rope optimization on the first feasible path instead of waiting for a better path, and (b) to get a first idea of how a shortened path compares with some path planning algorithms for the same processing time.

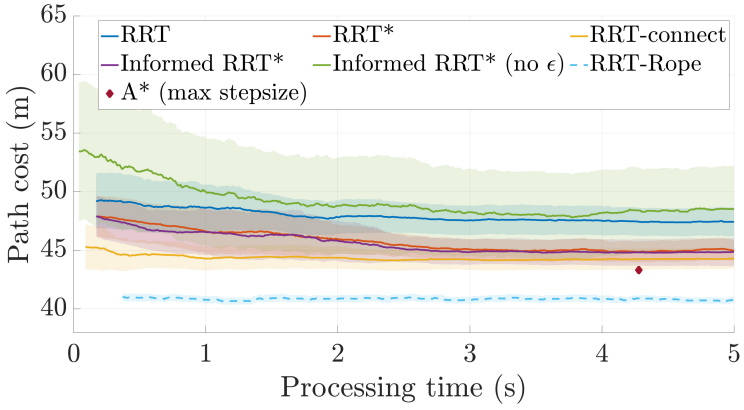
The first conclusion can be easily deducted from Figs. 3.6a, 3.6b, and 3.6c, with the RRT-Rope path cost being almost constant with respect to processing time and with a low standard deviation. Since the path cost does not depend on the initial path quality,



(a) Env. 1.



(b) Env. 2.



(c) Env. 3.

Figure 3.6 Moving average of 200 samples of RRT-Rope with $\delta = 0.6\ m$ (cyan), RRT [LaValle and Kuffner, 2000] (blue), RRT* [Karaman and Frazzoli, 2011] (orange), RRT-connect [Kuffner and LaValle, 2000] (yellow), Informed RRT* [Gammell et al., 2014] (purple), Informed RRT* without ϵ (green), and A* [Hart et al., 1968] (burgundy). The moving average of the standard deviation is shown in shade.

partially due to the homotopy of paths in our environments, online optimization is obsolete. It shows good confidence in the fact that shortening should be performed after the first feasible path is found, to improve computation time without deteriorating final path cost.

Note that RRT-connect denotes the classic implementation here, while its implementation in RRT-Rope is slightly different. The effect of this variation is noteworthy between the first path finding time of the two Informed RRT* curves.

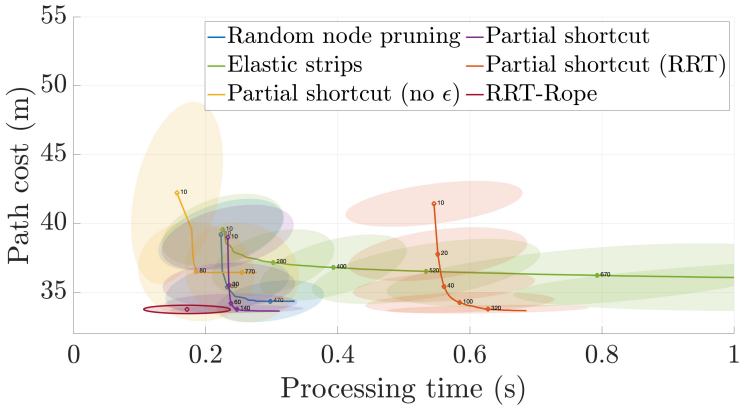
The path lengths confirm that RRT-Rope outperforms the other planning algorithms for a comparable processing time (i.e., the second objective). A* does not lead to the optimal path because of node resolution that was chosen as low as possible to quickly find the solution. More detailed surveys need to be performed to better understand when RRT-Rope outperforms path shortening algorithms.

3.5.2 Performance comparison

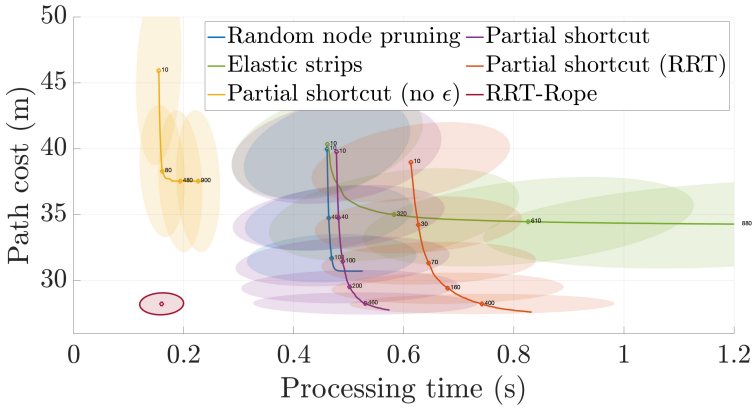
Fig. 3.7 shows the simulations carried out for 100 samples of each shortening algorithm considered. Unless explicitly stated in the legend, the shortening algorithms were performed on classic RRT-connect paths to find a first path in a fair amount of time.

The partial shortcut with RRT-connect without ϵ finds a first path more quickly but reaches a plateau far from the optimum because of the low node resolution. Random node pruning also reaches a plateau because of the node resolution which has become low after some nodes were pruned. As expected, elastic strips take much time to converge. Partial shortcut takes more time than RRT-Rope to converge because of irrelevant shortcuts and unnecessarily large node resolution as stated in section 3.1.

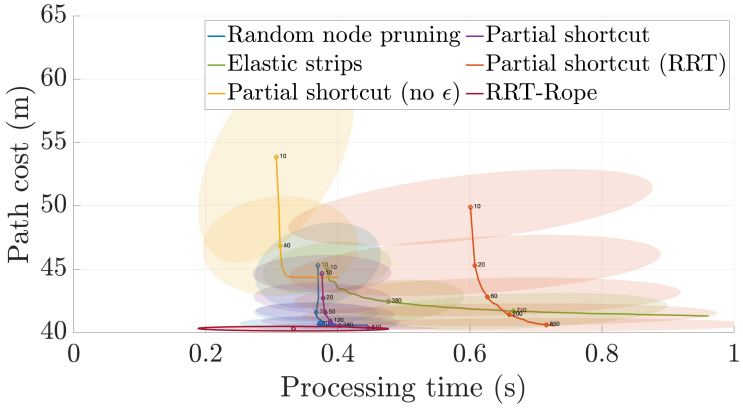
In Env. 2—that represent the classic stope environment— RRT-Rope error ellipse presents a big gap with other algorithms, and is 70% faster than the next best algorithm for the same path cost. This gap is reduced in Env. 1 that is narrower. In Env. 3—that was used to challenge the algorithm—the time spread is bigger because of the path finding that is less straightforward. In the end, RRT-Rope shows the best performance in each environment, with a path cost around 1.8% of optimum on average. It also offers the advantage of being a single point, giving the user the information of when the process is finished. In contrast, partial shortcut, i.e. the best second option, reaches its convergence after 150 iterations and 0.25 s in Env. 1, 500 iterations and 0.55 s in Env. 2, and 800 iterations and 0.45 s in Env. 3. It can be a downside to set the maximum number of iterations or processing time considering that convergence is difficult to evaluate online for an a priori unknown path.



(a) Env. 1.



(b) Env. 2.



(c) Env. 3.

Figure 3.7 Standard error ellipse plot for 100 samples of path shortening algorithms of RRT-Rope (burgundy), Random node pruning [Islam et al., 2012] (blue), Elastic strips [Brock and Khatib, 2002] (green), Partial shortcut without ϵ (yellow), Partial shortcut [Geraerts and Overmars, 2004] (purple), and Partial shortcut on RRT (red). The number of iterations is indicated next to the corresponding points.

3.5.3 Step-size sensitivity analysis

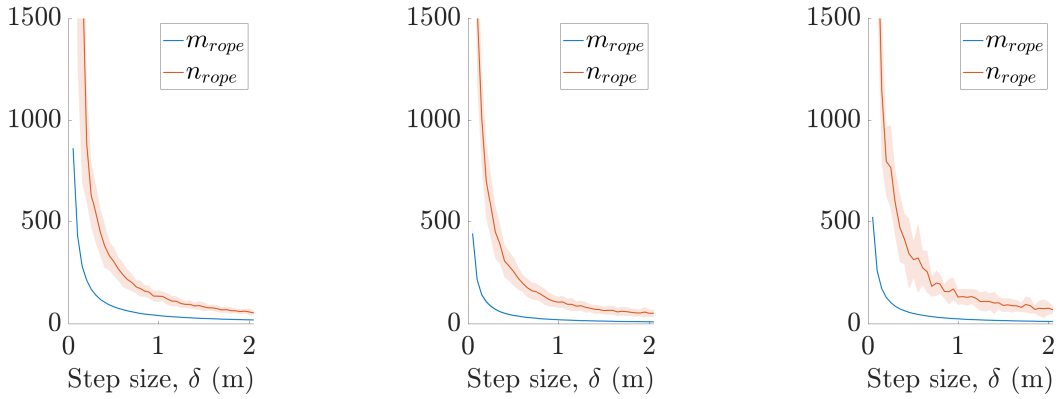
Parameter δ can be adjusted in RRT-Rope. This section evaluates the algorithm's sensitivity to this parameter by reporting results of path cost and processing time, in the same three environments, for values of δ between 0.05 and 2.05 m. Intuitively, an infinitesimally small value should give the path a perfect rope-like appearance, whereas a larger δ will transform the trajectory into a chain of longer links.

As shown in (3.2), the number of nodes in the tree follows approximately the ratio of the sum of the branch lengths over δ. The number of final tree nodes n_{rope} after RRT-Rope can be expressed as

$$n' \leq n_{rope} \leq n' + \frac{m\,(m+1)}{2}, \tag{3.5}$$

where n' is the number of tree nodes before optimization, and $\frac{m\,(m+1)}{2}$ is the upper bound of the path waypoints added during optimization, with m being the number of waypoints of τ_f previously defined in (3.3). $c(\tau_f)$, n, and the branch lengths can be regarded as independent of the path optimization step, since they only depend on the search algorithm. By substituting (3.2) and (3.3) into (3.5) and considering that δ is the only variable, (3.6) is obtained as follows:

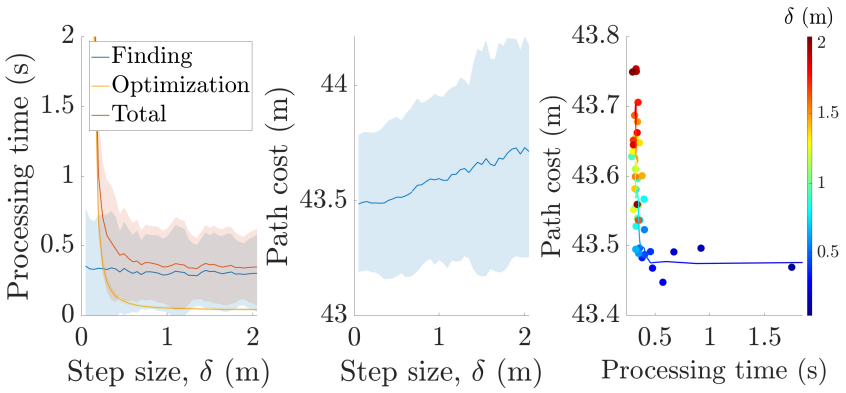
$$\frac{K_1}{\delta} \leq n_{rope} \leq \frac{K_2}{\delta} + \frac{K_3}{\delta^2}. \tag{3.6}$$



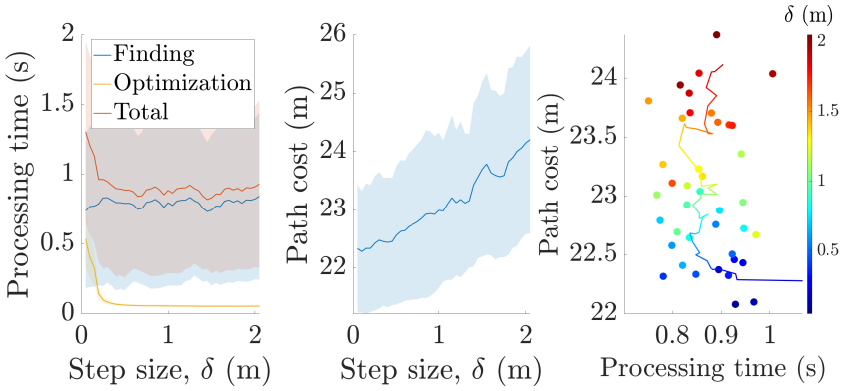
(a) Env. 1. (b) Env. 2. (c) Env. 3.

Figure 3.8 Number of path waypoints (blue) and tree nodes (red) for different δ values.

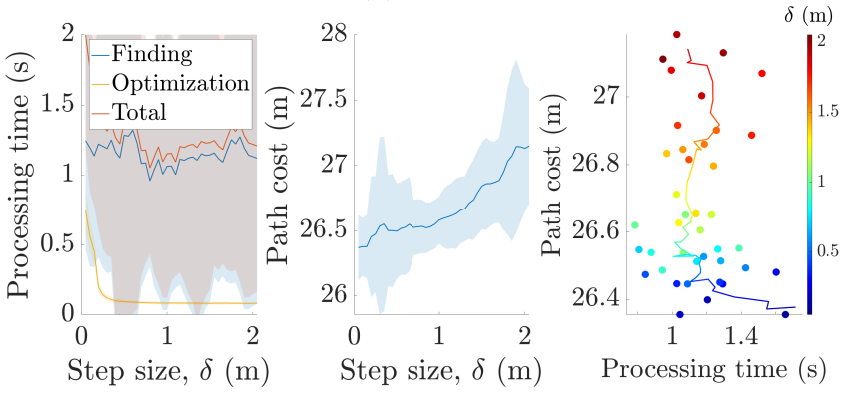
Fig. 3.8 shows n_{rope} in red, and the number of final path waypoints m_{rope} (defined in (3.4)) in blue. The red shaded area represents the standard deviation of n_{rope}, which is caused by (a) randomness of $c(\tau_f)$, n, and the branch lengths, and (b) variations inside the limit bounds of (3.6). The nearly nonexistent blue shaded area is due to variations of $c(\tau_{rope})$, which are shown to be quite low in Fig. 3.6 for a given δ.



(a) Env. 1.



(b) Env. 2.



(c) Env. 3.

Figure 3.9 Processing time (left), path cost (center), and total (right) for different δ values. The colored curve represents the moving average.

Also, variations in Fig. 3.9 show that for $\Delta\delta < 1.94\,\overline{\delta}$, the path cost variation $\Delta c < 0.08\,\overline{c}$. Therefore, the curve of path waypoints can be considered as following an inverse function of δ if Δc is neglected.

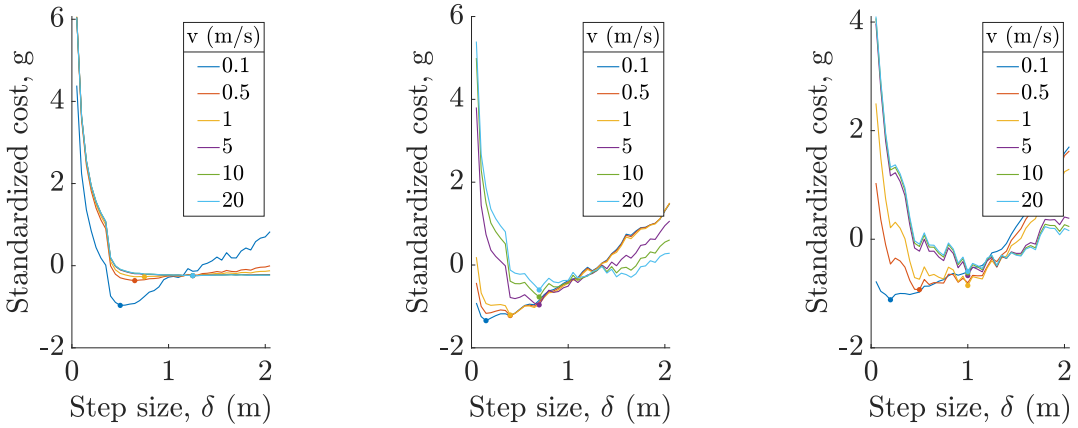
One can imagine that increasing the number of tree nodes by reducing δ would lead to a longer processing time. Fig. 3.9 shows the experimental processing time for path optimization in yellow, the total time in red, and the path finding time in blue. The path cost is represented in the center column of the figures. As expected, processing time decreases as a function of δ, whereas path cost grows with it.

The figure on the right side of Fig. 3.9 shows the same results on a cost-time axis. By looking at the moving average curve, we can see that δ generally enables us to make a trade-off between processing time and path cost.

To find an optimal δ minimizing the total time to move from q_I to q_G, a cost function of total flight time (including processing and travel) is introduced as

$$g_t = \frac{c(\tau)}{v} + t, \tag{3.7}$$

where v is the UAV velocity, t is the processing time, and $c(\tau)$ is the path cost.



(a) Env. 1. (b) Env. 2. (c) Env. 3.

Figure 3.10 Standardized cost g as a function of δ for different UAV velocities.

Fig. 3.10 shows the standardized value g of the cost function g_t for the sake of readability. Indeed, the flight duration is longer for slower UAVs, but one cares only about finding the value of δ minimizing the cost function. The minimum points of each curve show that a smaller value of δ is better for slow UAVs, whereas for faster UAVs it is preferable to have a lower computation time at the expense of a higher path cost. It is intuitively clear that a

fast vehicle has an interest in starting its movement quickly even if it has a slightly longer path to cover, whereas a slow vehicle has an interest in spending more time optimizing its route.

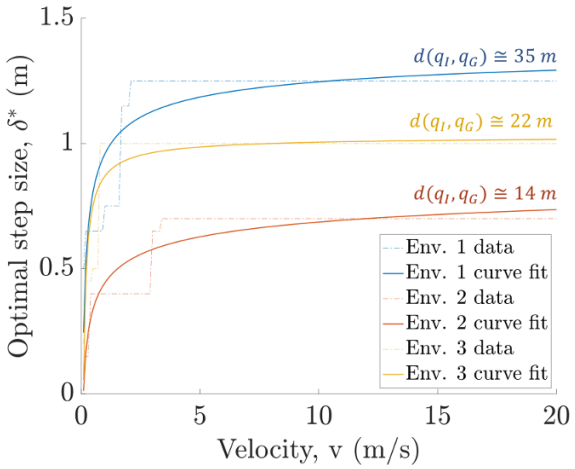


Figure 3.11 Value of δ^* optimizing standardized cost as a function of UAV velocities for different environments.

The minimum points δ^* are reported in Fig. 3.11. It can be observed that one should adopt a larger value of δ when the distance between q_I and q_G is higher. We also observed qualitatively that one should increase δ for a larger average cross-sectional area of the environment. However, this metric is more difficult to compute. Also, g has a relatively flat curve for each figure in the area where δ is between 0.5 and 1.5 m. This may motivate the user to set a constant value around 0.8 m for good performance in many environments.

3.6 DISCUSSION

Interestingly, pillar-like structures would not present a problem for RRT-Rope even if they would for a physical rope, as illustrated in Fig. 11 (right). In the same way, the algorithm could find a new homotopy in an uncluttered environment with wide passages. However, RRT-Rope could face local minima problems in complex environments, as illustrated in Fig. 3.12 (left). In mine stopes, this problem does not arise because the walls are smooth from the macro point of view.

The main challenge in resolving this hypothetical issue is detecting the local minima. A potential strategy for doing so is to use Informed-RRT* to escape local minima and to estimate (a) if the configuration has escaped a local minimum by monitoring the path cost decrease, and (b) if the algorithm has not yet converged thanks to the path cost decrease rate.

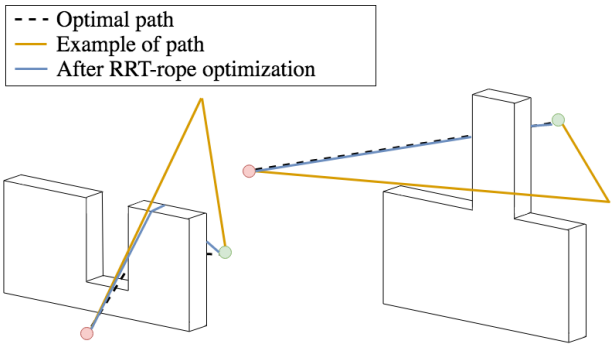


Figure 3.12 Representation of local minima situations (left) and non-problematic situations (right) with RRT-Rope.

The other approach to improving RRT-Rope is to include dynamic constraints in the problem definition. This could be done by detecting turns and imposing angle restriction constraints similarly to [Tian et al., 2019], or by using a local planner coupled with RRT-Rope.

3.7 CONCLUSION

The presented RRT-Rope algorithm can solve near-optimal path planning problems in a short amount of time for large environments. The algorithm's speed is inherited from an altered RRT-connect algorithm's fast time to compute a feasible path, coupled to a fast rope-inspired optimization algorithm for a random trajectory. Path finding time is improved by deleting infinitesimal step size which reduces random node generations needed in large uncluttered environments. The first path is sufficient for rope shortening algorithm which performance has been shown to be almost independent from the input path quality in 3-ball homotopic environments. Irrelevant shortcuts are avoided by checking farthest nodes first and detecting straight lines. Intermediate nodes insertion ensures near-optimality and uniform resolution. RRT-Rope produces an equal or shorter path than the other presented algorithms in a shorter computation time, and consistent behaviors without the need for a varying number of iterations thanks to deterministic selection.

The results are loosely sensitive to the environment and to the step size δ. This latter parameter can be optimized for a given path cost and robot velocity. Future works include introducing dynamic constraints and handling more complex environments.

CHAPTER 4

TAPE

Preamble

Title: TAPE: Tether-Aware Path Planning for Autonomous Exploration of Unknown 3D Cavities using a Tangle-compatible Tethered Aerial Robot.

French title: TAPE: Planification de trajectoire tenant compte du tether pour l'exploration autonome de cavités 3D inconnues à l'aide d'un drone filaire compatible avec les coincements.

Authors and affiliation:

L. Petit: Ph.D. candidate, Université de Sherbrooke, Faculty of Engineering, Department of Mechanical Engineering. Createk Design Lab.

A. Lussier Desbiens: Professor, Université de Sherbrooke, Faculty of Engineering, Department of Mechanical Engineering. Createk Design Lab.

Date of acceptance: 6 July 2022

Acceptance status: Final published version

Journal: IEEE Robotics and Automation Letters (RA-L)

Reference: [Petit and Lussier Desbiens, 2022]

Contribution to the document: The method leverages RRT-Rope results for quick path queries. This paper contributes to the thesis by elaborating an exploration algorithm that minimizes the path length while keeping the tether under a maximum length. This algorithm is able to run in real time due to its hierarchical architecture and it achieves the autonomy objective introduced in Section 1.4.4. The results show a better accuracy and precision on the maximum length of unwound tether than the other techniques compared. TAPE allows to improve the solution that meets the mission objectives 1 (RTH) and 3 (navigation to a waypoint chosen by the pilot) defined in Section 1.4, by taking into account the tether. In addition, the algorithm meets objectives 2 (display of relevant frontiers) and 4 (autonomous exploration).

Note: Following corrections requested by the members of the jury, the content of this article differs slightly from the one that was accepted.

Abstract: This paper presents the first method for autonomous exploration of unknown cavities in three dimensions (3D) that focuses on minimizing the distance traveled and the length of tether unwound. Considering that the tether entanglements are little influenced by the global path, our approach employs a 2-level hierarchical architecture. The global frontier-based planning solves a Traveling Salesman Problem (TSP) to minimize the distance. The local planning attempts to minimize the path cost and the tether length using an adjustable decision function whose parameters play on the trade-off between these two values. The proposed method, TAPE, is evaluated through detailed simulation studies as well as field tests. On average, our method generates a 4.1% increase in distance traveled compared to the TSP solution without our local planner, with which the length of the tether remains below the maximum allowed value in 53% of the simulated cases against 100% with our method.

Résumé français: Cet article présente la première méthode d'exploration autonome de cavités inconnues en trois dimensions (3D) qui se concentre sur la minimisation de la distance parcourue et de la longueur du fil déroulé. Considérant que les coincements du fil sont peu influencés par la trajectoire globale, notre approche utilise une architecture hiérarchique à deux niveaux. La planification globale basée sur les frontières résout un problème de voyageur de commerce (TSP) pour minimiser la distance. La planification locale tente de minimiser le coût du chemin et la longueur du fil en utilisant une fonction de décision ajustable dont les paramètres jouent sur le compromis entre ces deux valeurs. La méthode proposée, TAPE, est évaluée par des études de simulation détaillées ainsi que par des tests sur le terrain. En moyenne, notre méthode génère une augmentation de 4,1% de la distance parcourue par rapport à la solution TSP sans notre planificateur local, avec laquelle la longueur de l'attache reste inférieure à la valeur maximale autorisée dans 53% des cas simulés contre 100% avec notre méthode.

4.1 INTRODUCTION

Autonomous robotic exploration is a major field of research and has led to many uses in areas such as search and rescue [Balta et al., 2016], inspection [Bircher et al., 2015] and surveillance [Grocholsky et al., 2006]. The main objective of the exploration of a priori unknown cavities is to cover the environment completely, in the most efficient way possible. To do so, a robot is equipped with sensors, such as cameras or LiDARs. Indoor environments are often GPS-denied. The data obtained from the sensors is therefore used to build a map of the environment and perform the localization task with a Simultaneous Localization And Mapping (SLAM) approach.

In this work, we consider the problem of exploring three-dimensional (3D) cavities with a tethered aerial robot. Such spaces are often accessed by a narrow entrance tunnel and can be large-scale, posing challenges for autonomy, computation and communication. The latter is important to allow an operator to interact with the robot in real time, e.g. to inspect the wall of an underground mining stope. Tethered Unmanned Aerial Vehicles (UAVs) offer the advantage of unlimited flight time and no loss of communication.

It is desirable to have a rewindable tether to allow for proper 3D modeling and to avoid getting caught on the ground with a dangling wire. The tether is assumed to be always in taut condition. The extra wire should be stored in an on-board spool, allowing the UAV to continue its mission even if the tether gets caught in obstacles. However, its maximum length should be kept small since it directly affects the drone's payload at takeoff. Our testing platform is the NetherDrone (see Fig. 4.8a) which fulfills these characteristics except for the rewind which will be integrated in the future.

The approach presented in this paper aims at exploring 3D cavities by minimizing the global path cost and respecting a maximum length of available tether, thanks to a new hierarchical framework composed of a global level of exploration path planning that solves a Traveling Salesman Problem (TSP) and a local level of path planning whose parameters make it possible to adjust the trade-off between the tether unwinding and the path cost. On average, our method of tether-aware path planning for exploration (TAPE) generates a 4.1% increase in distance traveled compared to the TSP solution, with which the length of the tether remains below the maximum allowed value in 53% of the simulated cases against 100% with TAPE.

This paper is organized as follows. Section 4.2 defines the problem addressed. In section4.3, related work regarding autonomous exploration and navigation is described and the main differences are discussed. In section4.4, the proposed approach is presented and detailed.

Section 4.5 provides results from simulations in various environments, along with a real-world flight with a mapping UAV. A comparative study of the methods is also presented. Finally, this paper concludes with a discussion and potential perspectives.

4.2 PRELIMINARY MATERIAL

4.2.1 Assumptions

Any rigid 3D object pose can be defined by a 6-parameter configuration: $(x, y, z, \phi, \theta, \psi)$ and the group of motions is $SE(3) = \mathbb{R}^3 \times SO(3)$. An aerial robot used for environment mapping (e.g. a quadcopter) is likely to move at low speed, thus keeping the roll and pitch parameters (ϕ, θ) close to zero. Also, the yaw parameter (ψ) can be controlled independently. A 3D mapping robot should be equipped with an omnidirectional sensor (i.e. spherical field of view). If not, ψ should be controlled to let the mapping sensors point towards the next exploration goal at all times. The problem can thus be considered holonomic with negligible dynamic constraints. Therefore, for a fixed speed, the mission time is proportional to the distance traveled. Furthermore, after applying a relevant off-line expansion to the occupied space to produce the collision-free space, the problem can be solved as a point-like problem.

Sensor footprints, including LiDARs, typically stop at surfaces unlike e.g. medical ultrasound. Narrow pockets, whose exploration would result in a collision, can therefore not be covered. This results in a residual volume that is not connected to the simply connected set of collision free configurations.

We focus on exploration of cavities, such as mining stopes—whose volume calculation is crucial to optimize stockpile, excavation, and extension operations. They are likely to be 3-ball homotopic by the nature of their creation (i.e. by explosion). The entrance tunnel can be artificially closed to give a bounded volume (see Fig. 4.2). Environments that are not 3-ball homotopic will also be considered since we cannot guarantee the form of any unknown environment. Environments such as underground tunnel networks are outside the scope of this study and should be explored with more appropriate approaches such as [Dang et al., 2019, Cao et al., 2021b, Cao et al., 2021a].

In our case, the tether is anchored to a fixed base station located at takeoff position and is assumed to be taut at any time, as it is retractable. In some cases, entanglements (i.e. tether contact with an obstacle) are unavoidable to allow the exploration of an environment. Since the extra wire is stored on-board and does not restrict the movement

of the robot in case of a contact with an obstacle, entanglements are authorized, unlike the approach developed in [Shapovalov and Pereira, 2020b].

4.2.2 Notation

We define the world $\mathcal{W} = \mathbb{R}^3$ and consider a tethered aerial robot $\mathcal{A}$, defined in $\mathcal{W}$ as a point, and ξ_t the configuration of $\mathcal{A}$ at time t, with $\xi_t = (x_t, y_t, z_t)$.

Let $V \subset \mathbb{R}^3$ be a bounded volume to be explored. $\mathcal{M}$ is an occupancy map representation of all the observable space in V, discretized with cubical voxels $m \in \mathcal{M}$, and is incrementally built from observations of the omnidirectional sensor $\mathcal{S}$ with a maximum effective range d_{max}.

The search space V can be decomposed into the free space V_{free} and the occupied space V_{occ}. The simply connected set of collision free configurations is denoted by $\Xi \subseteq V_{free}$.

Let $\bar{\mathcal{V}}_m \subseteq \Xi$ be the set of configurations from which a voxel m can be perceived from $\mathcal{S}$, and $V_\xi \subseteq V$ the volume perceived by $\mathcal{S}$ at ξ.

The residual map $\mathcal{M}_{res} \subseteq \mathcal{M}_{unknown}$ has a volume $V_{res} = \bigcup_{m \in \mathcal{M}} (m \mid \bar{\mathcal{V}}_m = \emptyset)$. Considering $\mathcal{L} \subset \mathbb{R}^3$, the set of configurations along the robot past trajectory, the tether length will be denoted as $L(\mathcal{L})$ and the perceived volume at time t is $V_t = \bigcup_{\xi \in \mathcal{L}_t} V_\xi$.

The goal of the exploration is to identify the free and occupied parts of V. The explored volume can then be expressed as $V^E = V^E_{free} \bigcup V^E_{occ} = \bigcup_{\xi \in \mathcal{L}} V_\xi$.

4.2.3 Problem statement

Given an initially unknown bounded volume V, incrementally identified in a mapping representation $\mathcal{M}$, find a collision free path $\mathcal{T} : [0, 1] \rightarrow \Xi$ that passes through $\xi_1, \xi_2, ...$, such that

a) the boundary conditions are respected:

$$\begin{aligned} \mathcal{T}(0) &= \xi_{current}, \\ \mathcal{T}(1) &= \xi_{home}, \end{aligned} \tag{4.1}$$

b) the tether remains below a maximal length L_{MAX}:

$$\max(L(\mathcal{L} \rightarrow \mathcal{L} + \mathcal{T}))) < L_{MAX}, \tag{4.2}$$

c) the observable volume is fully explored:

$$\bigcup_{\xi \in (\mathcal{T} \bigcup \mathcal{L})} V_\xi = V \setminus V_{res}, \tag{4.3}$$

d) the path cost function $c : \Sigma_\Xi \to \mathbb{R}_{\geq 0}$, which assigns a non-negative cost to the set Σ of all paths in Ξ that respect (4.1)(4.2)(4.3), is minimized:

$$c(\mathcal{T}^*) = \min_{\mathcal{T} \in \Sigma_\Xi} c(\mathcal{T}). \tag{4.4}$$

When the environment is considered fully explored as in (4.3) or if the tether is insufficient for exploring the whole volume, find a collision-free path $\mathcal{T} : [0, 1] \to \Xi$ that respect (4.1)(4.2)(4.4) to return the robot to its takeoff location.

Due to the a priori unknown nature of the environment, the problem must be solved incrementally as the robot explores and the path must be computed on-line, with the latest sensor readings.

4.3 RELATED WORK

According to [Yamauchi, 1997], exploration can be defined as the act of moving through an unknown environment while building a map that can be used for subsequent navigation. This problem has been addressed by several approaches, the most effective of which can be grouped into 4 categories: topological exploration, information theory, random sampling-based methods, and frontier-based methods.

Topological exploration is based on the topological representation of the environment (e.g. with a connectivity graph based on map segmentation) [Thrun, 1998,Kim et al., 2013,Silver et al., 2006b]. With edge and intersection detection, it can be efficient in subterranean tunnel networks or offices by focusing on topological completeness. However, it is not suitable for large cavities, which often cannot be divided into separate sections, and which require map completeness (e.g. for volume calculation).

In information theory, the goal is to move the robot to maximize the amount of new information to expect for the next sensors readings [Bai et al., 2016, Tabib et al., 2016, Stachniss et al., 2005]. These methods are often greedy and global optimality is affected by their myopia.

Random sampling-based methods are based on growing a Rapidly-exploring Random Tree (RRT) [LaValle and Kuffner, 2000] or use a similar planner [Karaman and Frazzoli, 2011].

These methods include Next-Best-View Planner (NBVP) [Bircher et al., 2016] and Graph-Based exploration Planner (GBP) [Dang et al., 2019]. They are very effective in tunnel-like environments but often suffer from a) randomness and therefore neglect of partially explored areas, or b) greedy behavior when a reward gain is assigned, in the same manner as information theory.

Frontier-based methods rely on the idea to move to a boundary between V_{free}^{E} and $V_{unknown}$ [Yamauchi, 1997]. These popular exploration methods have been applied to many problems [Dornhege and Kleiner, 2011, Gao et al., 2018]. [Yamauchi, 1997] states that the main question in robotics exploration should be "Given what you know about the world, where should you move to gain as much new information as possible?". This method selects the nearest frontier to allow the robot to incrementally expand its map $\mathcal{M}$. Then, other ways of selecting the appropriate frontier were evaluated (e.g. the largest one, a viewpoint that maximizes the number of observable frontiers) [Juliá et al., 2012]. This approach however only partially apply to this work since we focus on an optimal exploration mission where we want all the information at the end but the intermediate information is not essential, except perhaps to optimize the rest of the mission. With these frontier selection methods, the problem is not solved in a globally optimal way in the sense of (4.4). As stated in [Juliá et al., 2012], the appropriate exploration algorithm depends very much on the problem. In this case, one should focus on the completeness of the map and not the speed of discovering the largest part of the map in a greedy way, as many algorithms do.

To optimize the overall path, some frontier-based approaches have considered the selection problem as a variant of the TSP [Kulich et al., 2011, Kulich et al., 2019]. Other methods such as Technologies for Autonomous Robot Exploration (TARE) also solve this problem for viewpoints [Cao et al., 2021b, Cao et al., 2021a]. TARE explores in a more complete and rapid way than the methods considered as state of the art such as NBVP [Bircher et al., 2016] and GBP [Dang et al., 2019]. However, they do not consider the return to home in the optimization of the exploration path once the map is fully explored, in order to satisfy (4.1). In tunnel-like or office environments, this is of little importance since the entire path is composed of round trips, but in the context of large cavity exploration, it is an important factor to consider.

All methods mentioned so far do not consider the case of a tethered robot, which must respect (4.2). Research has been done to solve a 2D TSP for given waypoints in a known space with the constraint of a non-entangling path. It is based on the observation of path homotopy classes thanks to their identification by the h-signature [McCammon and Hollinger, 2017]. This same identification was used in 2D tangle-free exploration with

the nearest frontier method [Shapovalov and Pereira, 2020b]. In 3D, an entanglement may or may not be present for the same homotopy class, so h-signatures are not relevant information [Bhattacharya et al., 2010]. Work has been done to model a tether taut in 3D around an obstacle [Xiao et al., 2018]. This model can be refined by incorporating multi-relaxation, and intermediate relaxation, as discussed in section 4.4.1.

Most successful exploration models do not consider the case of a tethered aerial robot. Those that do [Shapovalov and Pereira, 2020b] are not optimized in terms of exploration path length and only consider the two-dimensional case. Moreover, the 3D modeling of a tether is close to reality but not reliable in some cases. The goal of this work is to provide a new 3D exploration framework minimizing the mission time in cavities, while respecting a maximum tether length.

4.4 PROPOSED APPROACH

This section starts by giving a method to compute a 3D tether. Then, the steps for processing the frontiers are stated. Finally, the global and local planning strategy are explained.

4.4.1 Tether model

Algorithm 4: Tether model

Input: Map $\mathcal{M}$, trajectory $\mathcal{L}$, contact points CP, max tether length L_{max}
Output: Final tether length L, max tether length L_{max}, contact points CP

```
1  for every WP in L do
2      current_contact ← CP(n);
3      relax_flag ← false;
4      if CP.size() > 1 then
5          last_contact ← CP(n-1);
6          relax_flag ← not ObstacleConfined(current_contact, last_contact, WP);
7      if relax_flag then
8          do
9              CP.pop();
10             repeat lines 2-6;
11         while relax_flag;
12     else
13         if CheckCollision(current_contact, WP) then
14             CP.push(collision_point);
15         else if CP.size() > 1 then
16             RC ← RayCasting(current_contact, WP);
17             if not ObstacleConfined(current_contact, last_contact, RC(1)) then
18                 CP ← PredictTether(RC, CP);
19     L ← CP(n).cost() + dist(CP(n), WP);
20     L_max ← max(L,L_max);
21 return L, L_max, CP;
```

As said in section 4.3, the model defined in [Xiao et al., 2018] and explained in section 2.2.5 can be made more accurate by incorporating multi-relaxation, and intermediate relaxation. Algorithm 4 presents the model after these 2 modifications. The contact points CP must be initialized with the position of the base station. Multi-relaxation is implemented in lines 8 to 11 to deal with the case where several contact points are visible simultaneously for a given waypoint. Also, in 3D, it may happen that the triangle containing (current_contact, last_contact, WP) is not collision free, but the contact point should move in the direction where the robot pulls the tether. Lines 15-18 implement intermediate relaxation via a recursive call to ensure that the line (current_contact, WP) is followed as if it were waypoints, to find the new contact point.

4.4.2 Frontier processing

Similar to the 2D approach in [Yamauchi, 1997], frontiers can be extracted from a 3D occupancy map (OctoMap) [Hornung et al., 2013]. In this map, the voxels are classified in 3 categories: *free*, *occupied* and *unknown*. Frontiers are the voxels in $\mathcal{M}_{free}$ that have at least one neighboring voxel in $\mathcal{M}_{unknown}$.

Filtering

The voxels are filtered to satisfy the closed cavity assumption. It can be achieved with a box filter corresponding to the search space V. However, to ensure that only the entrance is artificially filtered and for the sake of generality (e.g. for concave environments), a best practice is to use a viewing frustum, as in the FrustumCulling filter of the PCL library [Rusu and Cousins, 2011]. The filter camera is oriented parallel to the ground, and looking backwards with respect to the home pose. The camera position $(x_{cam}, y_{cam}, z_{cam})$ and field of view can be determined at time $t = 0$. For a North-East-Down axis system, the dimensions provided by a vertical slice in $\mathcal{M}_{occ}$ at $x = x_0$ (shown by the red voxels in Fig. 4.1) allow to fix a) y_{cam} and z_{cam} at the center of this slice, and b) the dimensions of the near plane. Then, the constraint to cover, in the frustum, all the voxels of $\mathcal{M}_{occ}$ located in $x < x_0$, allows to fix x_{cam} as well as the field of view.

Clustering

In order to work with a reasonable number of potential views to evaluate, the frontiers are clustered using a Euclidean clustering algorithm in a Kd-tree structure [Rusu, 2010]. If the search space is likely to have too many clusters compared to the computational power available for the goal selection procedure detailed in section 4.4.3, it is preferable to use a k-means clustering algorithm.

Representation

Finally, a candidate viewpoint $\xi_{goal,i}$ is generated for the observation of each frontier cluster C_i. For the sake of computation time and without loss of performance for large spaces, $\xi_{goal,i}$ is generated as the closest point in Ξ to $m_{c,i}$, which is itself the closest point in C_i to the centroid of C_i. Note that the point $m_{c,i}$ is not necessarily equivalent to the centroid of the cluster. For the case of a concave cluster, the true centroid could belong to V_{occ} and/or not allow to observe C_i because of occlusions. The point $\xi_{goal,i}$, as defined above, allows to observe at least partially C_i. This is even more true when the clustering is done by euclidean clustering where the voxels are very likely to be neighbors, contrary to a k-mean algorithm where a cluster can be more scattered. In the case where C_i is not entirely visible from $\xi_{goal,i}$, the exploration completeness (4.3) is ensured by the fact that the voxels not visible in C_i will be taken into account for the next exploration step [Kulich et al., 2019]. For a user whose computation time is not a limitation, a visibility-based approach as in [Dornhege and Kleiner, 2011] can be used instead, but it is difficult to implement in real time.

4.4.3 Global path planning

With a large enough sensor range, it is possible to have a relatively good a priori information of the space to explore. Moreover, since cavities are simply connected topological spaces, exploring a tunnel or pocket that has not yet been perceived requires crossing one of the known frontiers to access it, and probably ultimately returning to it when the exploration of that area is complete. The distance traveled in that area is therefore often added to the travel distance, regardless of the order in which the frontiers are visited.

Under these assumptions, the solution that optimizes the path length of visiting the frontiers—given what you know at time t—is substantially close to the global solution and the problem can be viewed as a TSP. The only difference with a TSP is that the departure city (i.e. the drone's position) is not equivalent to the arrival city (i.e. the home position).

Other than that, the problem can be solved using state of the art algorithms. For a high number of clusters, computing an exact solution ($\approx \mathcal{O}(n^2)$) can be demanding but approximate algorithms give very close results in a reasonable time. The Lin-Kernighan heuristic is used to solve the TSP as it is one of the most powerful algorithms [Lin and Kernighan, 1973]. To increase the fidelity of the solution, the adjacency matrix used to solve the problem is based on the path cost between the different $\xi_{goal,i}$ positions. The cost can be computed with any path planner but RRT-Rope is preferred since it rapidly computes a near-optimal path in large uncluttered spaces [Petit and Lussier Desbiens, 2021].

Algorithm 5: Exploration path planner

Input: map $\mathcal{M}$, collision free space Ξ^E, positions ξ_{home}, $\xi_{current}$, ξ_{goal}
Output: exploration path $\mathcal{T}$

```
F ← ExtractFrontiers(M);
F.Filter();
if F.size() == 0 then
    T ← LocalPlanner(ξ_current, ξ_home, rth=true);
else
    C_{1,...,n} ← Cluster(F);
    ξ_{goal,1,...,n} ← Represent(C_{1,...,n});
    ξ_goal ← SolveTSP(ξ_{goal,1,...,n}, ξ_current, ξ_home);
    T ← LocalPlanner(ξ_current, ξ_goal);
return T;
```

The efficiency of the approaches presented in [Dang et al., 2019] and [Cao et al., 2021b] comes largely from their hierarchical framework. In 3D, as shown later in the section 4.5.1 by trying an approach to solving the TSP to minimize entanglements similar to [McCammon and Hollinger, 2017], the visiting order of the candidates in the TSP has little influence on the entanglements or the length of the tether, while the local behavior of the robot has a strong influence on these parameters. The approach presented here is therefore based on a global and a local level. The global planning takes into account the boundary conditions (4.1), the observation of all frontiers (4.3) and the minimization of the path (4.4) as a goal for solving the TSP, while the local planning takes into account the length of the tether (4.2) and the minimization of the path (4.4). This allows the method to run easily in real time as the tether is only considered locally. Replanning is performed at each map update to ensure path quality. Algorithm 5 presents the global exploration process. The function *LocalPlanner* is detailed below.

4.4.4 Local path planning

Once the TSP is solved, a goal candidate i is chosen and a path query is sent to the local planner to go to $\xi_{goal,i}$. The local path is based on the generation of 5 possible trajectories ($\mathcal{T}_G$, $\mathcal{T}_{HG}$, $\mathcal{T}_{CPG}$, $\mathcal{T}^t_{HG}$, $\mathcal{T}^t_{CPG}$) whose choice is determined by a decision function. These trajectories result from 2 steps: a) searching for path portions, and b) assembling and shortcutting. The path portions queries between 2 points can be done with any path planner but RRT-Rope [Petit and Lussier Desbiens, 2021] is preferred here. The procedure is detailed below.

Path options

$\mathcal{T}_G$ is the result of the path request from the drone's position $\xi_{current}$ to the next goal candidate ξ_{goal}. $\mathcal{T}_{HG}$ is obtained by searching 2 portions of path $\tau_1(\xi_{current}, \xi_{home})$, and $\tau_2(\xi_{home}, \xi_{goal})$. These portions are assembled and a deterministic shortcut as in [Petit and

Lussier Desbiens, 2021] is applied on the resulting path. $\mathcal{T}^t_{HG}$ is obtained in the same way but the shortcut applied is different. It is the same algorithm as for the calculation of the tether for given waypoints. Thus, the path obtained remains stuck in the local minima where the tether may get stuck. $\mathcal{T}_{CPG}$ and $\mathcal{T}^t_{CPG}$ are similar to $\mathcal{T}_{HG}$ and $\mathcal{T}^t_{HG}$ but take as input portions of path that pass through the contact points in the reverse order, i.e. $\tau_0(\xi_{current}, CP_n)$, $\tau_1(CP_n, CP_{n-1})$, ..., $\tau_n(CP_1, \xi_{home})$, $\tau_{n+1}(\xi_{home}, \xi_{goal})$.

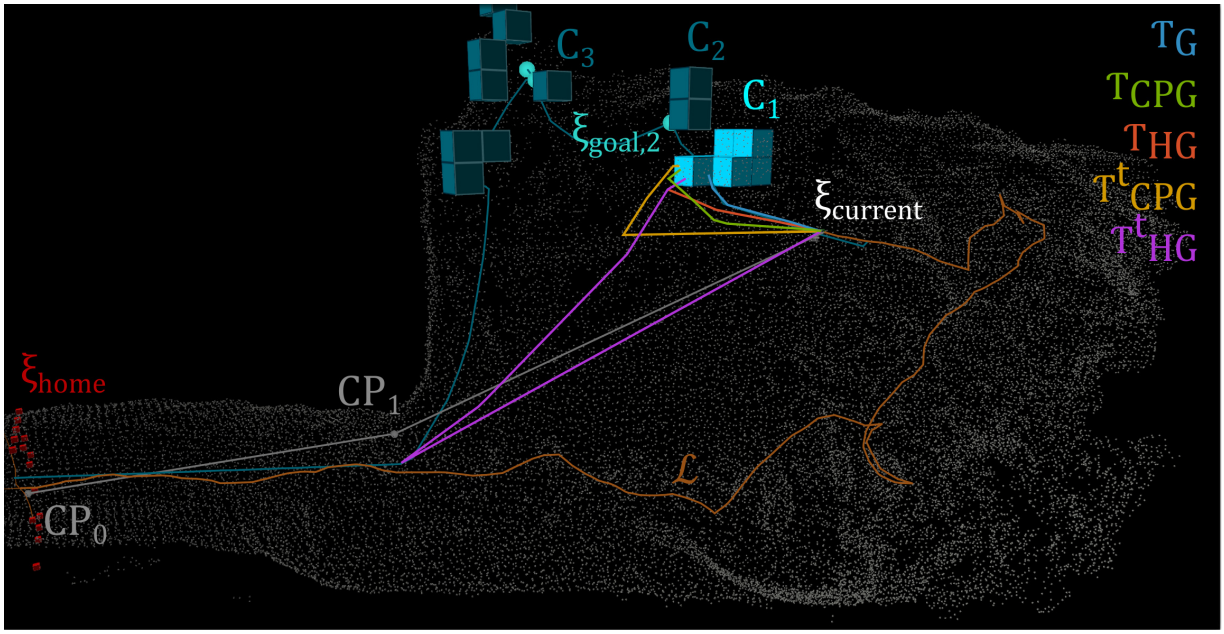


Figure 4.1 An example of exploration in Env. 1. The path options are in blue, red, green, yellow and purple. The past trajectory is in brown, the contact points and tether model in grey. The global exploration path and the frontiers clusters are in dark blue with the viewpoints and the next cluster in cyan.

For all paths, shortcuts are applied after assembly, and the assembly points (i.e. home or contact points) are not retained in the final path. The 5 possible paths are shown in Fig. 4.1. These paths lead to an informed range of options from the shortest ($\mathcal{T}_G$) to the one that allows the total untangling of the tether ($\mathcal{T}^t_{CPG}$), through options that allow to partially untangle the tether while remaining relatively short. If there is only one contact point in the current tether model (i.e. $CP = \{\xi_{home}\}$), $\mathcal{T}_{CPG}$ and $\mathcal{T}^t_{CPG}$ will not be evaluated. It is also trivial that for the case of a return to home, $\xi_{goal} = \xi_{home}$. Then, $\mathcal{T}_{HG}$ and $\mathcal{T}^t_{HG}$ will not be calculated.

Decision function

The decision function for a past trajectory $\mathcal{L}$ and a planned path $\mathcal{T}$ is defined as

$$\begin{aligned} f(\mathcal{L}, \mathcal{T}) = & \, k_1 \times c(\mathcal{T}) \\ & + k_2 \times max(L(\mathcal{L} \rightarrow \mathcal{L} + \mathcal{T})) \\ & + k_3 \times L(\mathcal{L} + \mathcal{T}) \\ & + k_4 \times CP(\mathcal{L} + \mathcal{T}), \end{aligned} \tag{4.5}$$

whose terms are respectively the path cost $c(\mathcal{T})$ (i.e. distance to cover), the maximum tether length between $\xi_{current}$ and ξ_{goal}, the tether length at ξ_{goal} and the number of contact points at ξ_{goal}. In practice, the computation of the function L and the associated CP is only performed for $\mathcal{T}$ since the CP associated to $\mathcal{L}$ are kept in memory. These parameters allow us to make an informed decision that takes into account the distance (k_1) and the maximum tether length (k_2), and also to locally relax the problem (k_3, k_4) to stay far away from L_{MAX} for future path queries. A sensitivity analysis for these parameters is performed in section 4.5.1.

Algorithm 6: Local path planner

Input: map $\mathcal{M}$, collision free space Ξ^E, positions ξ_{home}, $\xi_{current}$, ξ_{goal}, contact points CP, RTH flag rth
Parameters: k_1, k_2, k_3, k_4
Output: local path $\mathcal{T}$

1 $\mathcal{T}_G \leftarrow$ RRT-Rope($\xi_{current}$, ξ_{goal});
2 **if** CP.size() = 1 **and** rth **then**
3 **return** $\mathcal{T}_G$;
4 $\tau_2 \leftarrow$ RRT-Rope(ξ_{home}, ξ_{goal});
5 **if not** rth **then**
6 $\tau_1 \leftarrow$ RRT-Rope($\xi_{current}$, ξ_{home});
7 $\mathcal{T}_{HG} \leftarrow$ RopeShortcut(τ_1 + τ_2);
8 $\mathcal{T}^t_{HG} \leftarrow$ TetherShortcut(τ_1 + τ_2);
9 **if** CP.size() > 1 **then**
10 $\tau_1 \leftarrow$ RRT-Rope($\xi_{current}$, CP_n);
11 **for** $i \leftarrow 0$ **to** CP.size() **do**
12 τ_1.add(RRT-Rope(CP_{n-i}, CP_{n-i-1}));
13 $\mathcal{T}_{CPG} \leftarrow$ RopeShortcut(τ_1 + τ_2);
14 $\mathcal{T}^t_{CPG} \leftarrow$ TetherShortcut(τ_1 + τ_2);
15 $\mathcal{T}_{options} \leftarrow$ ($\mathcal{T}_G$, $\mathcal{T}_{HG}$, $\mathcal{T}^t_{HG}$, $\mathcal{T}_{CPG}$, $\mathcal{T}^t_{CPG}$);
16 **return** $\mathcal{T}_{options}$(MinElement(CostFunction($\mathcal{T}_{options}$)));

Algorithm 6 presents the approach used to compute a local path. Note that the function *RopeShortcut* uses only the shortcut described in [Petit and Lussier Desbiens, 2021] with a custom path to replace lines 1 through 3 of the RRT-Rope algorithm. Also, the function

CostFunction is based on (4.5) after evaluating the tether-related variables as in section 4.4.1. The function *TetherShortcut* uses the same tether model but for a planned path instead of a past trajectory, and the collision tests are performed in the collision free space Ξ^E instead of $\mathcal{M}_{free}$ in order to generate a valid trajectory.

4.5 RESULTS

4.5.1 Simulation studies

The approach presented in this paper was deployed on ROS (Robot Operating System) and tested on 4 different simulated environments that were reconstructed from real LiDAR scans. The simulated UAV shown in Fig. 4.8a is equipped with a rotating Ouster LiDAR and has a quasi-omnidirectional field of view after a full rotation. A SLAM approach called RTAB-Map [Labbé and Michaud, 2018] is used to provide localization data and the OctoMap from which the frontiers can be extracted. Simulations were performed with an HP Z440 Workstation with 12 Intel Xeon processors running at 3.5 GHz, 24 GB of RAM, with a node frequency of 10 Hz for the global planner and 15 Hz for local path planning.

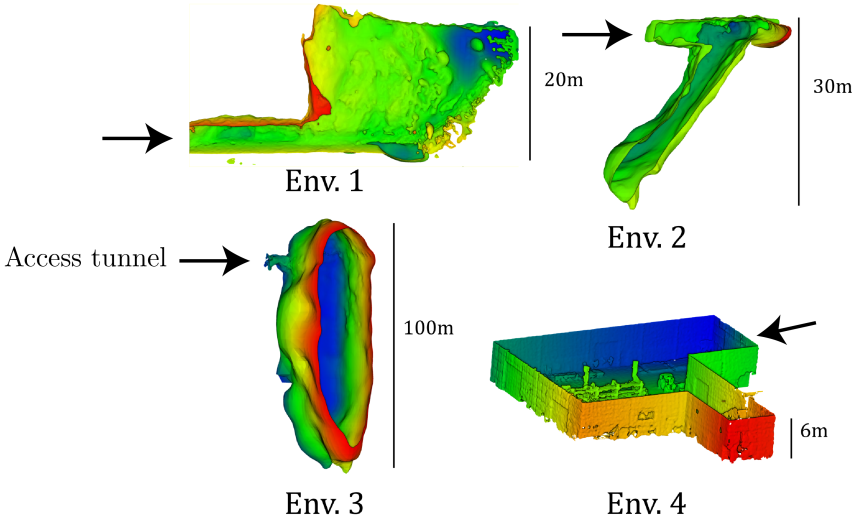


Figure 4.2 Cross-section view of the simulation environments.

The 4 environments pictured in Fig. 4.2 have been chosen to best represent the spatial characteristics of some typical exploration missions. *Env. 1* and *Env. 2* represent classic underground mining stopes with an access tunnel and narrow passages to navigate in order to map certain pockets. *Env. 2* is a bit narrower while *Env. 1* extends on an inclined plane at the end of the tunnel. *Env. 3* is also an underground mining stope with an access tunnel but is much larger, with some pockets in the walls. *Env. 4* is a cluttered indoor environment with shelves, pipes and other obstacles, to challenge the performance of the algorithm in environments with high risk of tether entanglements.

Performance comparison

Fig. 4.3 shows the results of the comparison of different algorithms: the selection of the biggest frontier in blue, the nearest frontier in red, the nearest frontier with our local planner in yellow, the resolution of a TSP in purple, our algorithm in green, and the resolution of a TSP taking into account the tether similar to [McCammon and Hollinger, 2017] without our local planner in cyan and with our local planner in burgundy. Table 4.1 gives an overview of the three different concepts studied in the compared algorithms. The first two algorithms are mainly used as an order of magnitude since it has already been proven that the TSP produces a shorter path [Kulich et al., 2011]. The other state of the art algorithms are not compared since such studies already exist [Juliá et al., 2012], and there is none to our knowledge that takes into account the tether for a 3D path.

Table 4.1 Exploration algorithms characteristics.

	Global planner (TSP)	Global planner (TSP with tether prediction)	Our local planner
Biggest			
Nearest			
Nearest*			X
TSP	X		
TAPE	X		X
gTAPE		X	
gTAPE*		X	X

In all environments, the algorithms that use our local planner do not exceed L_{MAX} in 100% of the cases on 32 samples and the objective (4.2) is therefore achieved.

As announced in section 4.4.3, *gTAPE* results in tether lengths similar to the other methods because the global path has little influence on entanglements. Also, it often gives a longer path because a) it sometimes leads to local hesitations between several successive iterations and b) it only takes into account the use of one planner to move between the frontiers because it would be too computationally demanding to consider the 5 path options for the $N \times (N-1)/2$ branches of the graph. With our local planner, *gTAPE** also suffers from this global distance increase.

In *Env. 1* and *Env. 3*, the 2 algorithms using the TSP (i.e. *TSP* and *TAPE*) are the shortest. In *Env. 2*, the path lengths are very similar between the TSP algorithms and the greedy algorithms, since the environment is quite narrow and the paths are likely to be similar. In *Env. 3* and *Env. 4*, we notice that our local planner leads to a slight increase in distance traveled when comparing *nearest* and *TSP* with *nearest** and *TAPE* respectively.

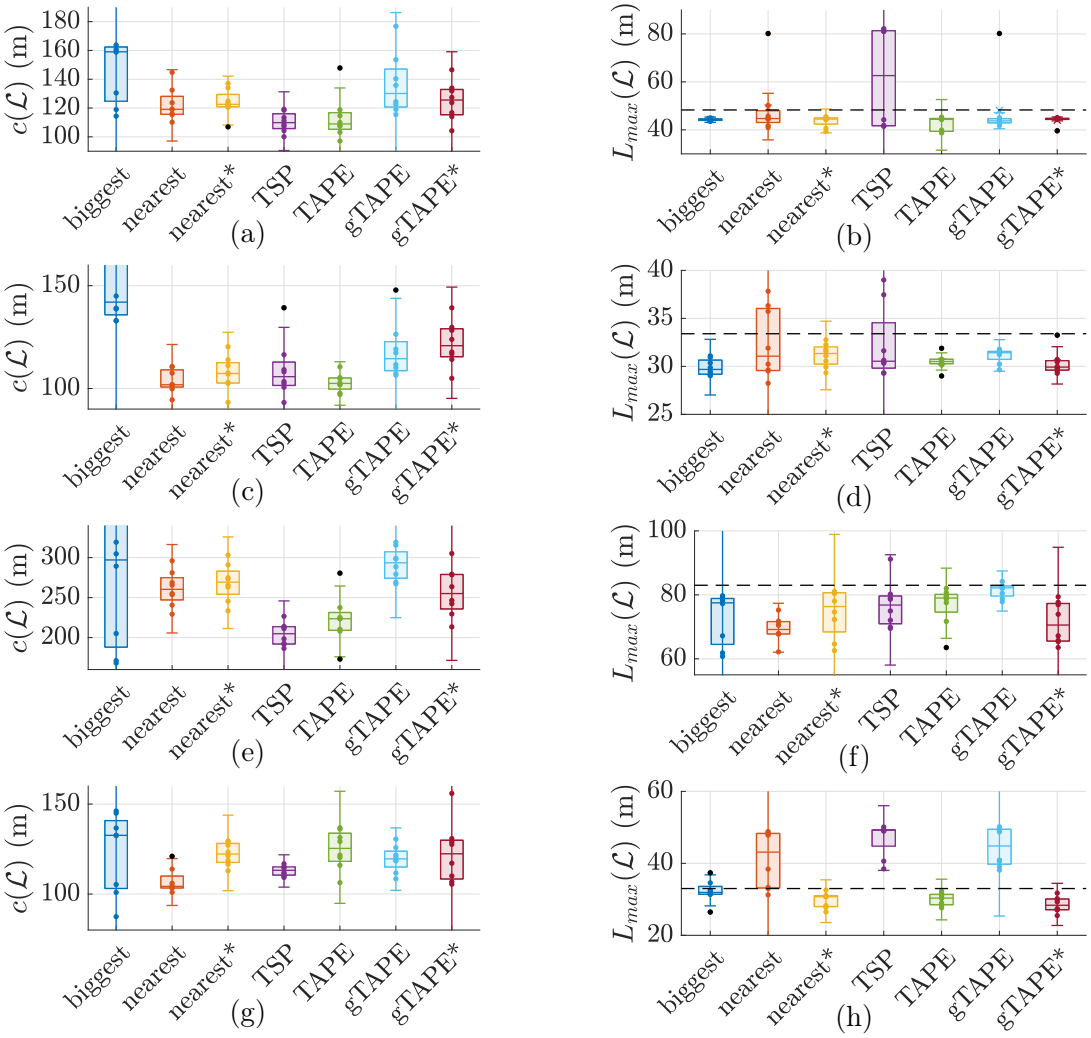
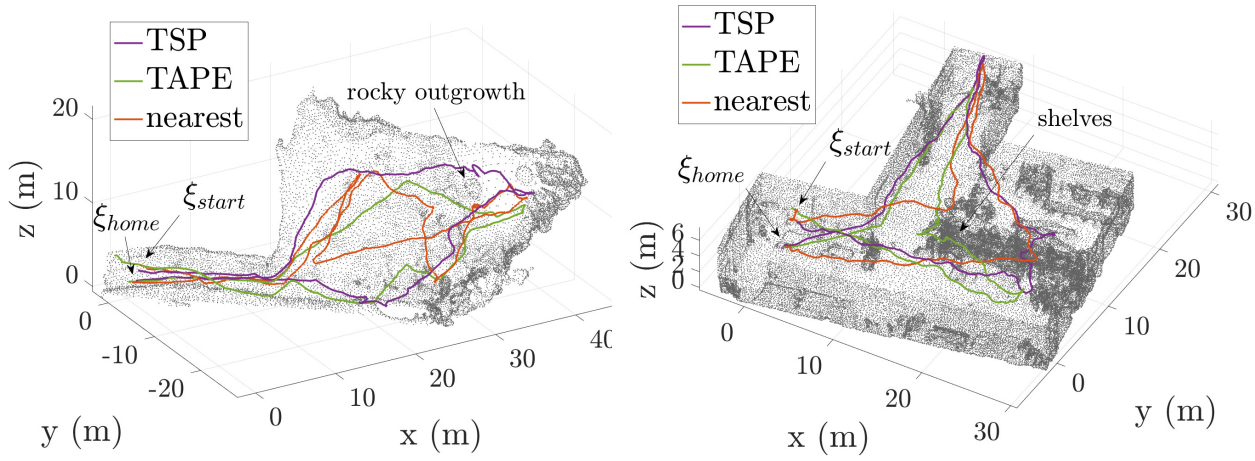


Figure 4.3 Comparison of covered distance and maximum tether length for different algorithms in a-b) Env. 1, c-d) Env. 2, e-f) Env. 3, g-h) Env. 4. The dotted line shows the maximum tether length L_{MAX} needed to travel the longest distance diagonally to the end of the cavity from the base station, taking the shortest path. The data are presented as a boxplot for 8 samples in each category, with the cross representing the average, and the black dots representing the outliers. The parameters k_i used in (4.5) are (1, 7, 1, 10).

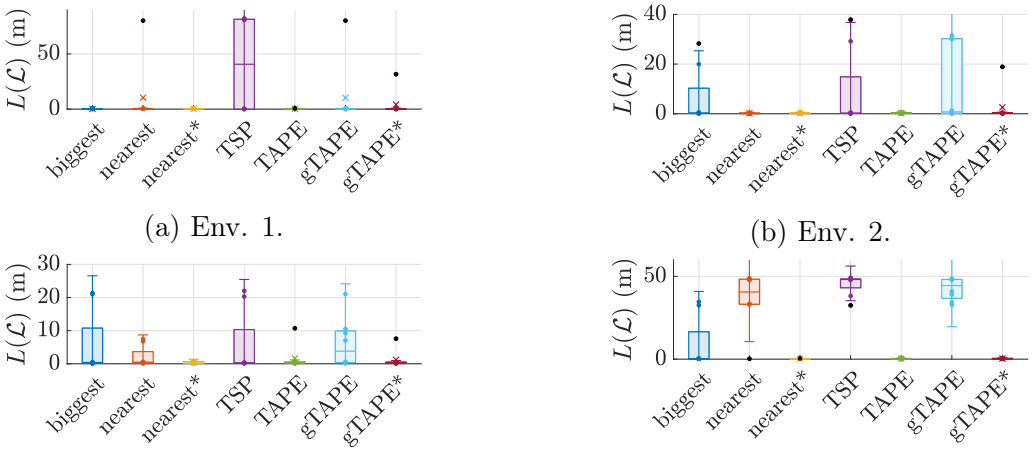
Fig. 4.4 shows 3 paths taken at random from the same dataset, for the greedy algorithm, the TSP and our algorithm. In Fig. 4.3g, we notice that the increase in distance avoids entanglements in the shelves located in the middle of the environment, as can be seen in Fig. 4.4b by observing the obstacle bypassing by our algorithm compared to the TSP. This behavior is also observed in Fig. 4.4a where our planner allows to go around the rocky outgrowth located on the top right. On average, our method generates a 4.1% increase in distance traveled compared to the globally optimal method (TSP), which entangles the tether in 46.9% of cases. In Fig. 4.3g, we observe the limits of the TSP for global path optimization, as the possibility of discovering a frontier from behind after bypassing a column is not well treated. It is in the limit of the case of a 3-ball homotopic environment, as explained earlier in the section 4.2.1. In *Env. 1*, *Env. 2* and *Env. 3*, a comparison of the distance traveled show the usefulness of using the TSP over the greedy algorithm to optimize the global path.



(a) Env. 1. (b) Env. 4.

Figure 4.4 Exploration path display.

Finally, a feature that was not stated in the initial objectives is the possibility to recover the tether after the mission. In the case of an unsafe or inaccessible environment for humans, a stuck tether results in a loss of material. Fig. 4.5 shows the final length of the tether at the end of the exploration mission for the same samples as in Fig. 4.3. Our method recovers the tether completely in 96.9% of the cases and results in a loss of only $10m$ in the only outlier case among the 32 trials.



(a) Env. 1.

(b) Env. 2.

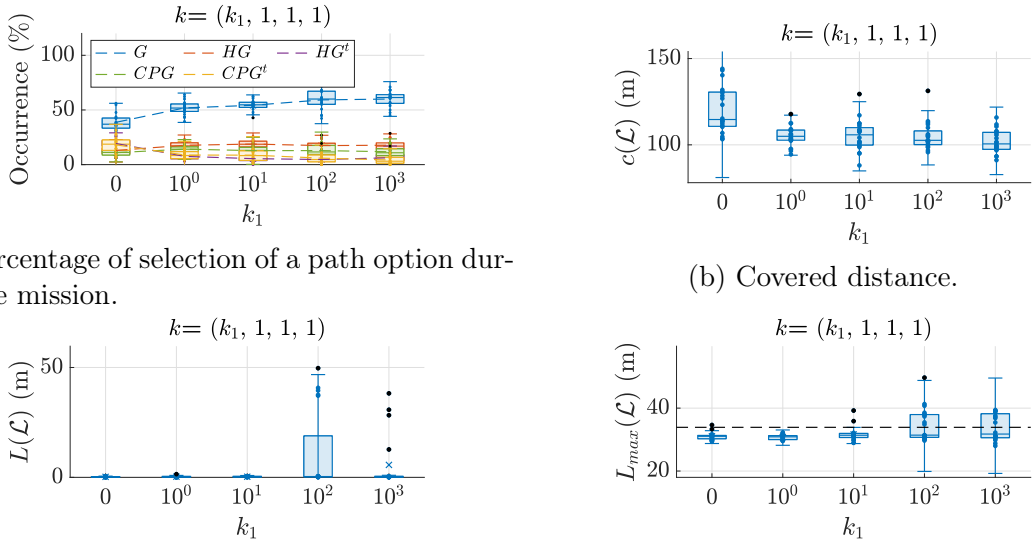
(c) Env. 3.

(d) Env. 4.

Figure 4.5 Tether length L at the end of the exploration mission.

Parameters sensitivity

Fig. 4.6 and Fig. 4.7 presents the results of simulations performed with different sets of parameters $k_{1\to4}$ in (4.5), with the aim of confirming their importance and giving the user a preliminary idea of how to choose them. The tests are performed in *Env. 2* as it is the smallest, with 20 samples per parameter set. This is where we will see a greater sensitivity in the parameters because the distances to be traveled and the tether lengths are shorter and one term of (4.5) can more quickly become important compared to another.



(a) Percentage of selection of a path option during the mission.

(b) Covered distance.

(c) Final tether length.

(d) Maximum tether length.

Figure 4.6 Results of the parametric analysis in Env. 2, for different values of k_1.

Fig. 4.6 shows the effects of variations of k_1. Fig. 4.6a shows that for a large k_1, $\mathcal{T}_G$ is chosen more often and the paths with a tether-shortcut less and less. The effect is a decrease of the path cost (Fig. 4.6b), and an increase of the final and maximum tether length (Fig. 4.6c and Fig. 4.6d). In our case, $k_1 = 1$ allows to decrease the path cost without a consequent increase of the tether length.

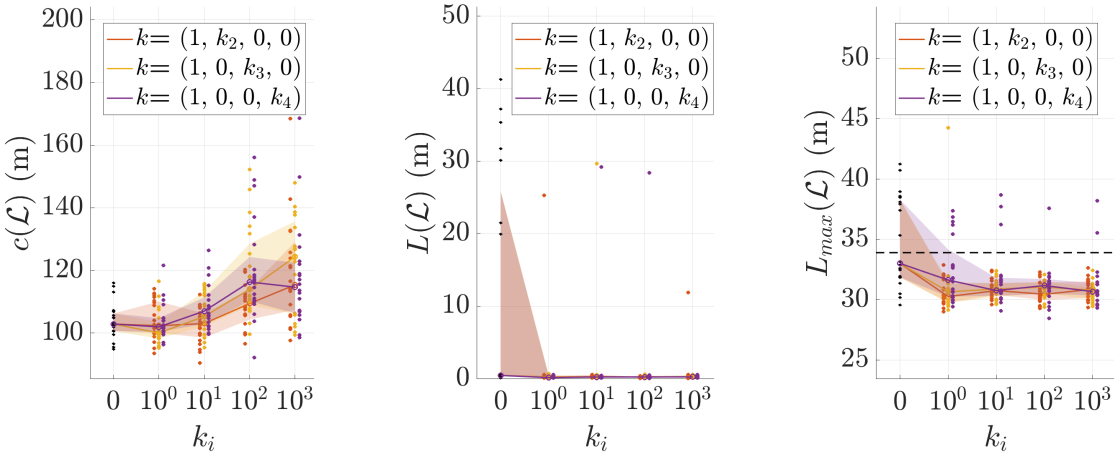
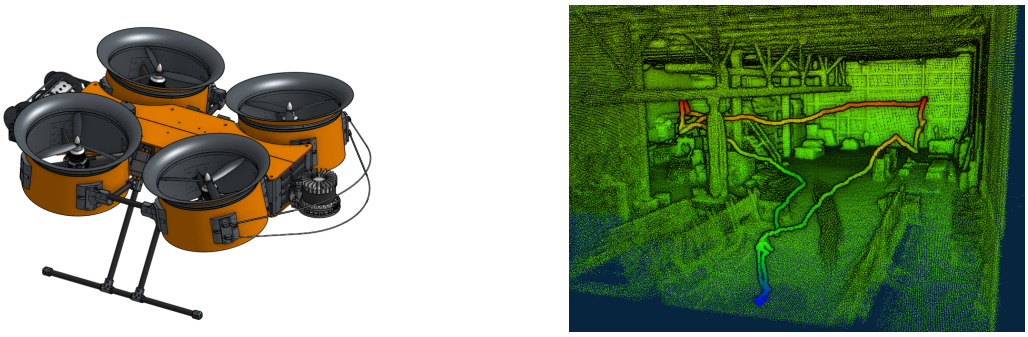


Figure 4.7 Results of the parametric analysis in Env. 2, for different values of k_2, k_3, k_4: a) covered distance, b) final tether length, c) maximum tether length.

Applying the same approach, we see that $k_{2\to4}$ each increase the path cost (Fig. 4.7a). k_2 significantly affects c only for $k_2 \leq 10$, decreases L and L_{max} by preventing entanglements when $k_2 \geq 1$. k_3 affects c a lot, decreases L_{max}, and $L \simeq 0$ for $k_3 \geq 10$. k_4 affects significantly c, decreases L_{max}, and $L \simeq 0$ when $k_4 \geq 100$. A set around $k = (1, 10, 1, 10)$ is thus a good choice for our case. A high precision is not necessary since a small variation of the parameters does not influence the results very much.

4.5.2 Experimental evaluation

Finally, our method is used by the NetherDrone (see Fig. 4.8a) to explore an unknown indoor environment. Fig. 4.8b shows the result of this flight. The UAV explores a volume of 6000 m^3 in 164 s, covering a distance of 74 m with a maximum tether length of 39 m. For operation safety reasons, a part of the environment on the left was marked as a no-go zone with a box filter. The results show that the drone follows a path that avoids the tether getting stuck between obstacles, while keeping this path fairly short.



(a) NetherDrone 3D render. (b) Exploration path display.

Figure 4.8 Real-world flight experiment.

4.6 DISCUSSION

A remaining limitation of the tether model is that collisions are only considered with obstacles detected by the sensor and not with the prediction of the tether itself, but this rare case did not occur among all our observations, thanks in part to the fact that entanglements are minimal with our approach.

A UAV that does not respect the taut tether assumption can do well with our algorithm if the voxel resolution is adapted to the environment complexity. In our test with the NetherDrone, we observed on the ground that the unrolled tether on the return path was always located within at most $1m$ of the unrolled tether on the outbound path, and it can then be easily rewound. This may reach its limits in complicated environments where the computation will become too expensive, or the mapping inaccuracies too noticeable.

Also, our method performs very well in cavities but might reach its limits for a large maze environment, where another algorithm than RRT-Rope [Petit and Lussier Desbiens, 2021] can be used for path generation but the integration in real time becomes challenging.

Regarding adaptability, it has been shown that the global path has little influence on the tether entanglements but is important to minimize the path distance, and the local planner can thus be integrated with other approaches such as GBP or TARE, instead of the frontier-TSP.

4.7 CONCLUSION

In this work, a hierarchical framework for path planning for 3D cavity exploration by a tethered aerial robot is proposed. The global planning focuses on minimizing the distance traveled by solving the Traveling Salesman Problem for clusters of frontiers. The local planning aims at minimizing the distance while avoiding tether entanglements, according to a decision function whose parameters allow to adjust the tether length and path cost

trade-off. Our method is evaluated in simulation and in a physical environment against other exploration methods, using the NetherDrone equipped with a rotating LiDAR. Our two-stage architecture is the first to our knowledge to focus on 3D exploration with a tether. The results confirm that it performs better in satisfying path minimization while respecting an acceptable deployed tether length. On average, our method generates a 4.1% increase in distance traveled compared to the method without our local planner, with which the length of the tether remains below the maximum allowed value in 53% of the simulated cases against 100% with our method. Future work plans to deploy this algorithm in more real-world environments.

CHAPTER 5

INTEGRATION

The previous chapters have set up the technical tools to perform the tasks that belong to the scope of this project. This chapter presents the integration of these tools into a complete system and returns to the mission objectives mentioned in Section 1.4. Some specific adjustments to the project are also presented since the methods (RRT-Rope and TAPE) were presented in a generic way to benefit the largest number of people in the robotic community. Finally, the performance of the system is evaluated during simulation missions and in real flight.

5.1 System configuration

This section presents the system configuration that incorporates the navigation tools presented in this work. Fig. 5.1 shows the system architecture.

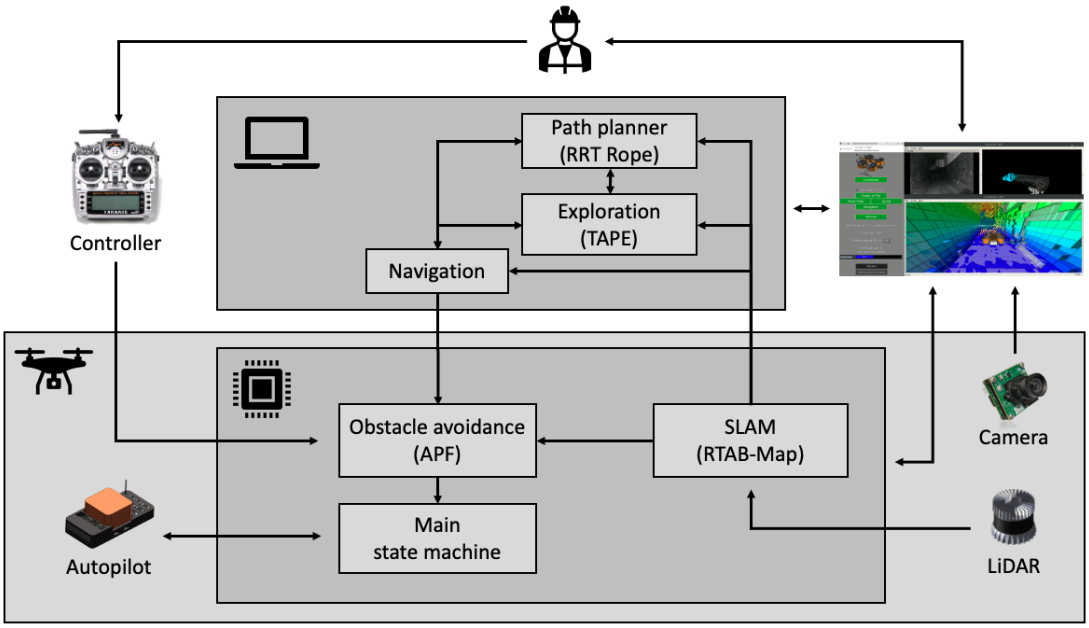


Figure 5.1 System architecture. The software blocks of the base station computer are in the grey box at the top. It communicates with the visual interface and with the onboard computer of the drone, through the tether. The light grey box at the bottom shows the hardware blocks on the drone, and the dark grey box shows the software blocks of the onboard computer. The pilot communicates with the drone through the joysticks of the remote control for manual teleoperation, and through the interface.

The pilot communicates with the system through a user interface, accessible via a laptop, and a remote control. He can therefore remotely operate the drone during an exploration mission. The drone is equipped with an onboard computer (PICO-WHU4: Intel i7-8665UE, 1.7GHz, Ubuntu 18) computer that communicates with the *Pixhawk PX4* autopilot through a state machine that is the main code for executing physical tasks (e.g., arming, taking off, sending engine commands in flight). The onboard computer runs *ROS* [Quigley et al., 2009], with a SLAM node that receives data from the Ouster LiDAR. During teleoperation, commands are sent to the obstacle avoidance system (APF), as described in [Dozois, 2022]. During autonomous operations, the autonomous navigation is generated on the laptop and the data is exchanged with the SLAM and APF nodes via the tether. The navigation node is responsible for long term planning while obstacle avoidance is responsible for modulating the trajectory according to real-time sensor readings.

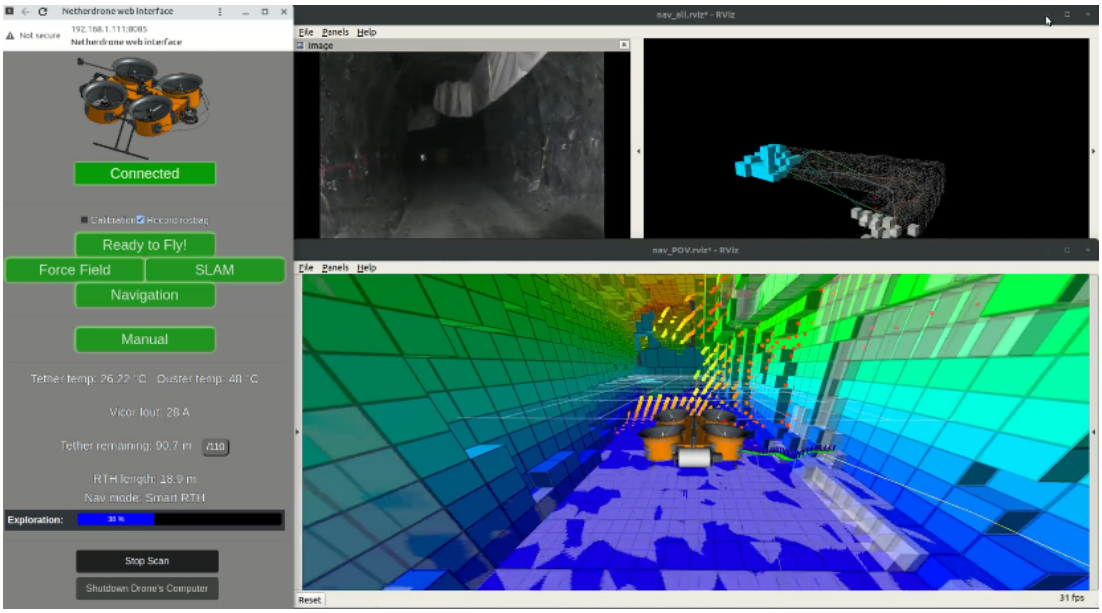


Figure 5.2 User interface. The interactive part is on the left. Information is displayed such as the temperature of the LiDAR, the length of the unwound tether, and the percentage of exploration completed. At the top middle, the camera view is relayed. RViz simulates a third person view (at the bottom) and a global view (at the top right), with indications such as the next frontier to visit, the global and local path, the past path and the tether model.

Fig. 5.2 shows a screenshot of the interface during the mission. The interactive part is located on the left. The pilot can, for example, arm the drone, take off, and turn off the onboard computer. Information is displayed such as the temperature of the LiDAR, the length of the unwound tether, and the percentage of exploration completed (the distance completed compared to the total distance calculated with TAPE). For the inspection of

the walls, the drone has an onboard camera whose images are displayed in the top middle. The *RViz* viewer [Kam et al., 2015] simulates a third person view (at the bottom) and a global view (at the top right), with indications such as the next frontier to visit, the global and local path, the past path and the tether model. The visual interface is a key element for the predictability of the system, so that the operator is informed of the intentions of the robot, via the display of the short term path and the next waypoints to reach. In addition, global information such as the exploration path, frontier clusters and the tether model improve the interpretability of the system for the operator who better understands the reasoning behind the algorithm decisions.

Fig. 5.3 shows the deployed system in a mining stope. To enable testing of algorithms without risking large operating costs, software-in-the-loop simulations can also be performed via the *Gazebo* simulation software [Koenig and Howard, 2004].

Figure 5.3 Operation of the NetherDrone in an underground mine.

5.2 Features

The TAPE and RRT-Rope algorithms create useful features for autonomous exploration and are suitable for different applications. Some project-specific challenges remain after the integration of these tools. This section presents the tweaks that are needed to fulfill the project tasks defined in Section 1.4 with the NetherDrone. As a reminder, the tasks are:

1. Return To Home (RTH),
2. exploration assistance,
3. semi-autonomous navigation,
4. autonomous exploration.

5.2.1 Return To Home

The RTH is calculated with TAPE and the path is sent to the navigation node in Fig. 5.1. Even without performing autonomous exploration, the RTH path computed with TAPE allows to minimize the tether length unlike RRT-Rope. If only the shortest path is preferred, RRT-Rope can be used instead. For an efficient RTH, a path follower and a path smoothing systems have been integrated. With these building blocks, the RTH can be triggered and performed at any time during the mission. These blocks are detailed below.

Path follower The navigation node integrates a path follower system that transfers waypoints one by one to the APF to achieve the given path. This path follower runs through a list of waypoints, the first of which is the goal and the last is the starting point. While the path follower is running, if one of the points is within a given distance from the drone (e.g., 1 meter), all subsequent points in the list are deleted. The last point in the dynamic list is then sent to the APF and the operation is repeated. To ensure good visibility of the points on the drone path with the LiDAR, the yaw is adjusted to point to the next waypoint at all times [1]. It ensures to covers the shadow areas located behind the drone due to its own body.

Path smoothing To avoid power spikes that are problematic for the power converter of the tethered drone, and reduce the accelerations that can be rough on the mechanisms, the drone must avoid aggressive maneuvers. Since RRT-Rope computes a path for a holonomic problem, the final trajectory must be smoothed. The overly aggressive maneuvers are

1. Except when the waypoint is too close to the drone in the horizontal plane (e.g., less than 40 cm) to avoid aggressive rotations when the drone is flying a near vertical path.

mainly due to yaw variations. Therefore, an Infinite Impulse Response (IIR) low-pass filter is incorporated into the yaw waypoint tracking in the APF. The yaw angle (ψ) depends on the angle pointing to the next waypoint (ψ_{ref}) via the following relation:

$$\psi_i = (1 - \alpha) * \psi_{ref} + \alpha * \psi_{i-1},$$

where α is a coefficient that depends on the sampling frequency and the desired cutoff frequency.

5.2.2 Exploration assistance

The second feature is an exploration assistance. This is performed using the frontier method which is integrated in TAPE. The frontiers are identified in blue in the global view in Fig. 5.2. They are also visible at the end of the tunnel in the third person view. This indicates to the pilot the remaining areas to explore. In TAPE, the frontiers are filtered and grouped, and the most relevant cluster to visit first, for the optimization of the global path, is highlighted to the user. In the tunnel, there is only one cluster after filtering and it is therefore highlighted but further on, Fig. 5.7 shows more clearly the highlighting compared to the other clusters. The gray voxels represent the filtered frontiers, according to Section 4.4.2. This filtering process eliminates the frontiers located in the tunnel axis behind the takeoff point, via a frustum culling. The frontiers that are detected in RTAB-Map but not connected to the free configuration space (i.e., incorrect data) are also filtered using a flood fill algorithm.

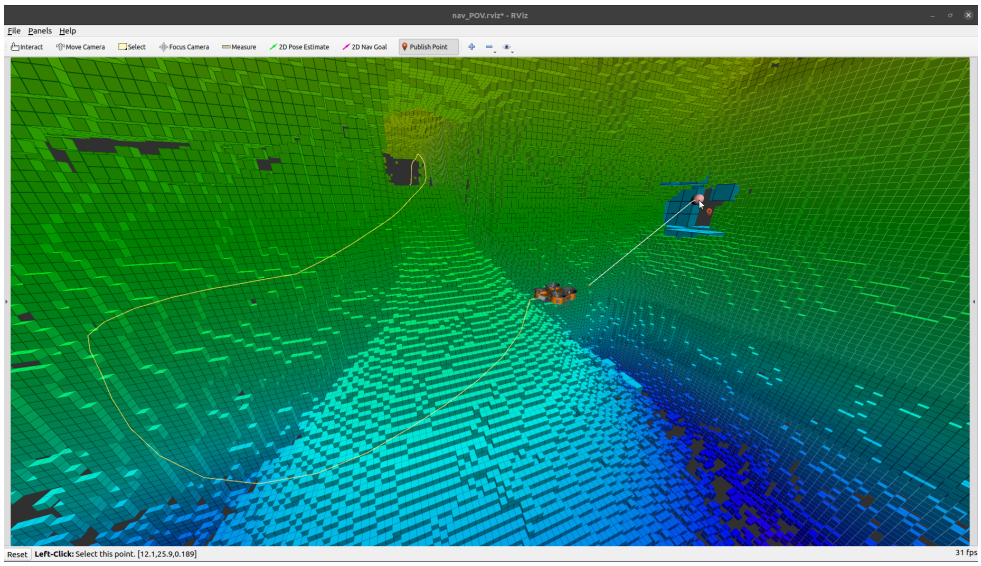


Figure 5.4 Manual point selection in RViz. The mouse of the user is in the upper right corner and the pink sphere is the corresponding goal (p_2). The planned path is in white.

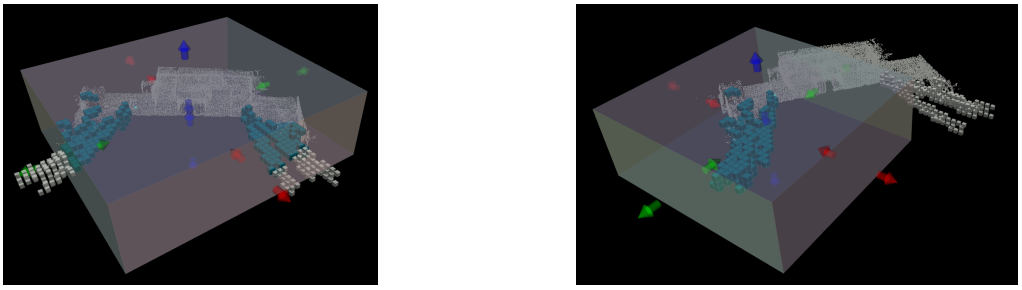
5.2.3 Semi-autonomous navigation

Using the same strategy as for the first task, a path can be requested to any reachable point with TAPE, or RRT-Rope if the tether is not considered. A feature has been implemented in the project to navigate to a point p_1 that the pilot can select by clicking at the desired location in RViz (global or third person view). Fig. 5.4 shows this feature. The point p_1 is found by ray casting among the visible voxels. As it is either a frontier or an obstacle, a point p_2 is generated by a Kd-tree [2] search as the closest point to p_1 in the free configuration space. This new point p_2 is then used as the goal of the desired path.

5.2.4 Autonomous exploration

The TAPE algorithm allows for intelligent exploration of cavities. Nevertheless, full autonomy is often complicated to deploy. Several adjustments specific to the project were implemented to make this system completely autonomous, including desirable features, computational time optimizations, and robustness adjustments (e.g., to exit from corner cases encountered in test missions carried out during the project, or to protect against external perturbations such as localization errors). These tweaks are listed below.

Bounded space In the case where the operator only wants to explore a portion of the space, for safety or relevance reasons, a dynamic bounding box is integrated into the viewer. As shown in Fig. 5.5, the pilot can change the size of the box in all three axes using the colored arrows. The frontiers are then filtered in accordance with this box.



(a) (b)

Figure 5.5 Exploration bounding box. The frontiers are shown in blue, and the filtered frontiers in gray.

Residual volume As mentioned in Section 4.2.1, unreachable frontiers are filtered since they belong to the residual volume that cannot be mapped. The detection of these unreachable frontiers is triggered by a voting result for a large number of failed path requests,

2. A Kd-tree is a binary search tree (BST) of dimension k, with k=3 here.

as stated in Section 2.3.3, or when they remain in the frontier state after a full rotation of the LiDAR when the drone has reached the desired viewpoint.

Frontier clusters limit To limit the number of path requests necessary to build the adjacency matrix, and the number of cities in the TSP, the number of clusters is limited to a maximum number (20 in our case). The smallest clusters are merged with the closest cluster. The inaccuracies created by this method are erased by the sequential effect of the exploration and the representation of the clusters presented in Section 4.4.2, since the unvisited frontiers will be taken into account at the next iteration with the new sensor readings.

Multi-resolution empty space To limit the time to search for points in the free configuration space (e.g., nearest points for cluster representation, or random point generation for RRT), the point cloud is represented as a multi-resolution space. When traversing the OctoMap, empty points are represented by the largest empty voxel, with a maximum size limit. Thus, points close to the walls are represented with a high point density where accuracy is important while points in the middle of large spaces are represented in a more sparse way where accuracy is not important. This allows to manage the memory more efficiently, similarly to the octree structure, and thus improving speed, at the cost of a small loss of precision.

Adaptive octree depth The resolution of the octree is also dynamically adapted with the environment. In the case of large environments where the computation of an exploration instance is expensive, the tree depth is decreased by one level to always respect the real-time computing condition. This results in a small reduction of precision, often insignificant in the large environments in question.

Local search tree for narrow spaces In the case of large stopes (e.g., 150 m deep and 70 m wide), some tight passages (e.g., 40 cm wide in the space of free configurations) slow down the search for a path with RRT without infinitesimal step size ϵ. To improve robustness of the path planning system, this situation is detected if the size of one of the two trees is less than a given number (e.g., 30 nodes) after a chosen number of iterations (e.g., 200). This occurs when the tree fails to find a collision-free branch that passes through the narrow passage due to the large number of irrelevant sampling points. When this is detected, the new node sampling operation is performed among the points located within a distance (e.g., 5 m) from the last node of the RRT.

Unknown space navigation Another approach for speeding up path planning time, that also sometimes reduce the path length, is to allow segments that cross the unknown space. As stated in [Quin et al., 2021] and [Wettach and Berns, 2010], sensor measurements are taken while the robot is moving so that the unknown space becomes a known space before the robot passes through it. The robot therefore remains in free space at all times.

Path tracking for tether modeling Since the map provided by the SLAM is dynamic and may contain errors, and the tether configuration depends on the past path, this path is retraced to verify that the points belong to the free space. If not, the points are moved to the nearest free space with a ray casting constraint between them, in order to allow for tether relaxation and increase robustness to external SLAM errors.

5.3 Performance overview

With all the adjustments mentioned above, the drone is equipped with a fully autonomous exploration system. This section presents data related to the efficiency of the system in order to highlight the calculation time, the robustness of the system, the quality and completeness of the map. Other key metrics are also given, such as mission time, length of tether unwound and distance traveled.

Table 5.1 Navigation system parameters.

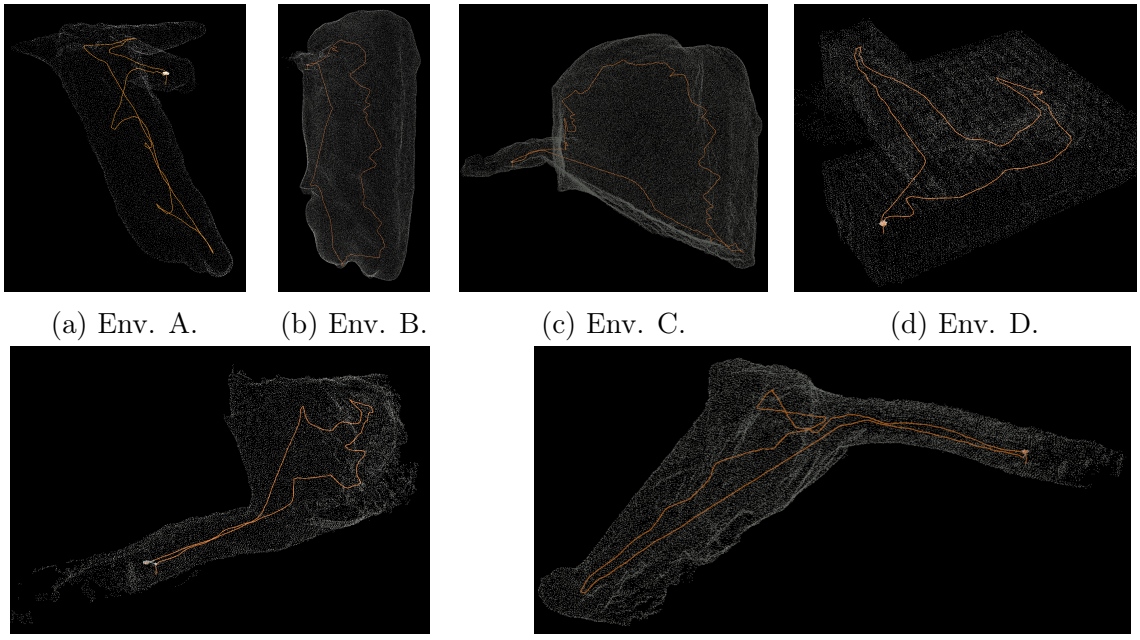
Parameters	Values
RRT-Rope intermediate nodes distance (δ) [m]	0.6
Obstacle ray tracing increment [m]	0.2
Obtacle radius offset [m]	1.2
Octree leaf voxel size [m]	0.3
Initial octree depth	16
Max. octree depth	14
Path shortcut distance [m]	1.3
Waypoint reached distance [m]	0.9
Goal reached distance [m]	1.2
Home reached distance [m]	0.5
Home land offset [m]	1.5
Path smoothing coefficient (α)	0.75
Min. waypoint horizontal distance for yaw tracking [m]	0.4

Table 5.1 gives the values used for the navigation parameters mentioned in this thesis. These parameters were chosen empirically. For example, the obstacle radius offset is large enough to avoid colliding with walls depending on the radius of the drone, but small enough to avoid virtually clogging the free configuration space in a narrow tunnel. Also, the home land offset must be greater than this value to allow a successful RTH request. Other parameters depend on the values of the flight controller such as the saturation

speed which set the waypoint reached distance to ensure a constant cruising speed. Finally, parameters such as δ and the octree depth were chosen as a good balance between accuracy and computation time.

The initial research question of this project is: *How to enable 3D autonomous exploration of unknown underground mining stopes using a tethered drone to provide a complete map while keeping the unwinding of the tether under the maximum allowed length and minimizing mission time?*

Chapters 3 and 4 have established that the system is able to minimize the distance traveled—which directly influences the mission time—and respect the unwinding condition of the tether, which allows to finish the mission and ensure the completeness of the map. The last aspect to consider concerns the real-time computing ability, i.e., the second factor that influences the mission time. Simulations of the global system have been performed in 6 environments whose scans are provided in Fig. 5.6 and Fig. 5.7.

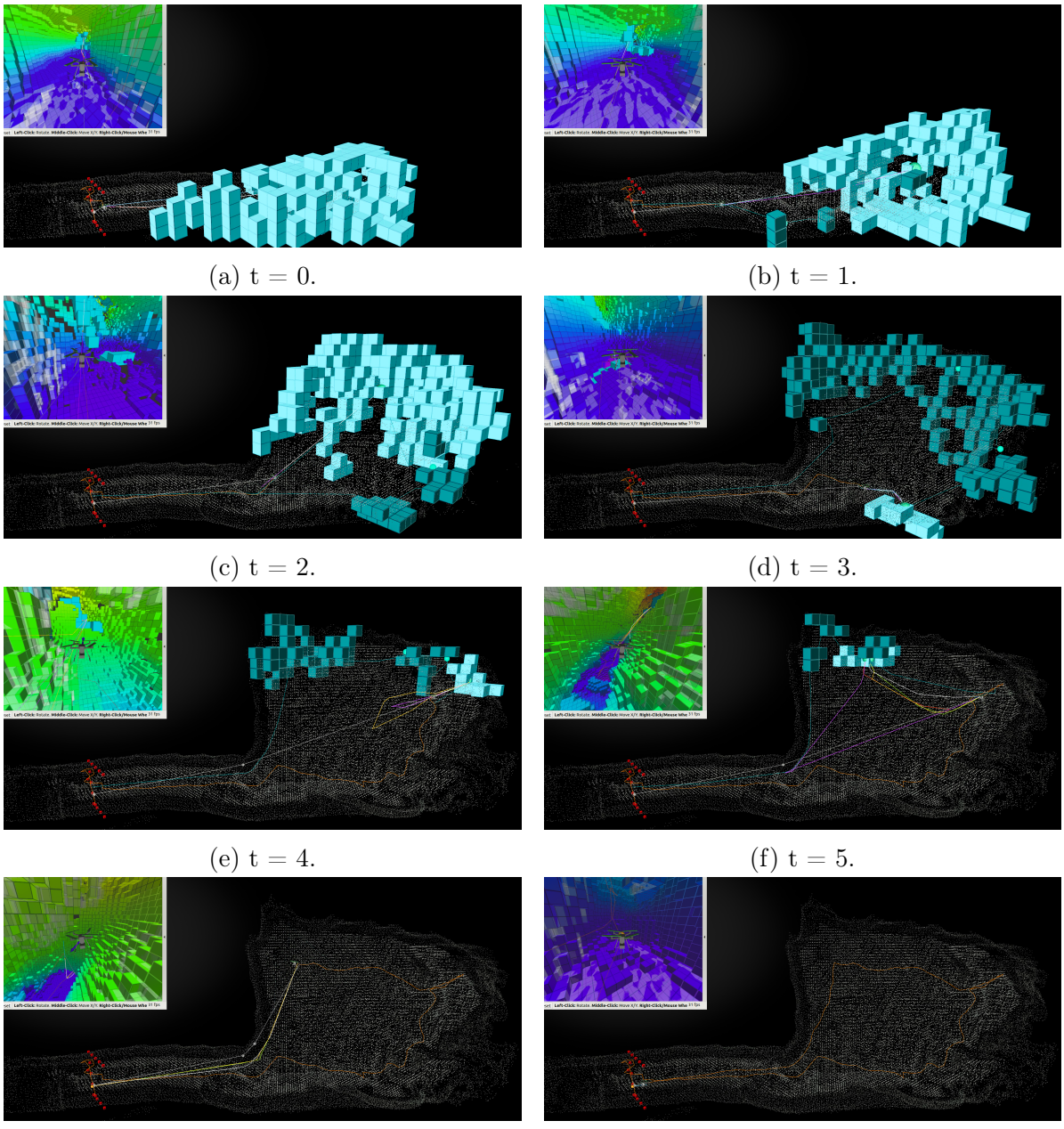


(a) Env. A. (b) Env. B. (c) Env. C. (d) Env. D.

(e) Env. E. (f) Env. F.

Figure 5.6 Exploration tests with the exploration system in 6 different environments.

Env. A is a small stope of 40 m depth, with 3 tunnels above and a diagonal cavity, and a width that varies between 2 m and 8 m. Env. B is a large stope of 50 m width and 130 m depth. Env. C is a large stope of 120 m of side. Env. D is a warehouse of 30 m side and 8 m height with several obstacles. Env. E is a narrow diagonal stope of 40 m length with a 3 m wide tunnel at the entrance. Env. F is similar with 2 m width in some places.



(a) t = 0. (b) t = 1.

(c) t = 2. (d) t = 3.

(e) t = 4. (f) t = 5.

(g) t = 6. (h) t = 7.

Figure 5.7 Exploration sequence in Env. E. The path options are in blue, red, green, yellow and purple. The past trajectory is in brown, the contact points and tether model in grey. The global exploration path and the frontiers clusters are in dark blue with the viewpoints and the next cluster chosen in cyan. The third-person view is in the top left corner.

Table 5.2 presents the results of an exploration mission in these 6 environments, with the same computer as described previously in Chapters 3 and 4. The mission time is less than 10 minutes in all the environments. The largest environments (Env. B and C) are challenging for computation time but even the maximum instance of TAPE ran in less than the targeted 4 seconds, i.e., the map update time. The average time of a TAPE instance is between 0.03 and 0.6 seconds, which fully meets the required criteria.

Table 5.2 Exploration metrics in 6 different environments.

	A	B	C	D	E	F
Covered distance [m]	114.1	292.4	363	126.9	111.2	180.8
Mission duration [min]	3.6	8.3	9.8	4.0	3.8	5.3
Max. tether length [m]	32.1	85.4	89.4	31.3	43.6	65.1
Avg. RRT-Rope computation time [3][s]	0.003	0.007	0.041	0.003	0.008	0.017
Max. RRT-Rope computation time [s]	0.016	0.170	0.368	0.027	0.030	0.080
Avg. local planning computation time [s]	0.034	0.077	0.237	0.028	0.046	0.041
Max. local planning computation time [s]	0.085	0.235	0.804	0.043	0.096	0.111
Avg. adjacency matrix computation time [s]	0.057	0.196	0.614	0.085	0.119	0.196
Max. adjacency matrix computation time [s]	0.139	2.582	3.675	0.284	0.670	0.455
Avg. TSP computation time [s]	0.001	0.006	0.006	0.05	0.001	0.002
Max. TSP computation time [s]	0.004	0.099	0.070	0.035	0.007	0.012
Avg. TAPE computation time [s]	0.029	0.203	0.621	0.084	0.119	0.131
Max. TAPE computation time [s]	0.140	2.589	3.775	0.309	0.676	0.468

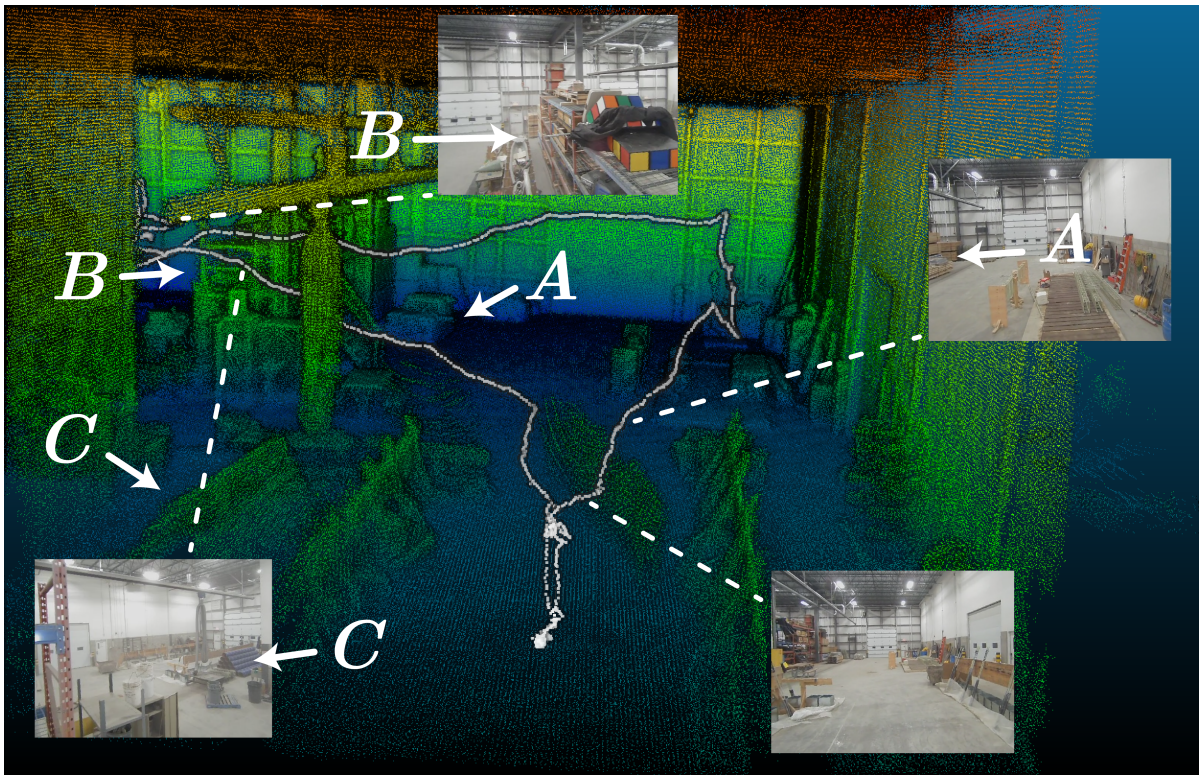
The results highlight that the limiting factor for the total time is the adjacency matrix computation time, i.e. the computation of a path between all the cities of the TSP, and the local planning time, which is also influenced by the tether model computation. If the path planning time were 3 times longer—i.e., the performance of the next best algorithm of the state of the art, after RRT-Rope (in Fig. 3.7b)—the real-time computing condition would not be respected and the mission time would go from below 10 to about 20 minutes in the largest environments considered. Improvements for the adjacency matrix computation time and local planning time are discussed in the next chapter.

Regarding the robustness of the system, more than 1,000 autonomous explorations were performed, in simulation, in these 6 environments without any corner case error, leading to all missions being successfully completed.

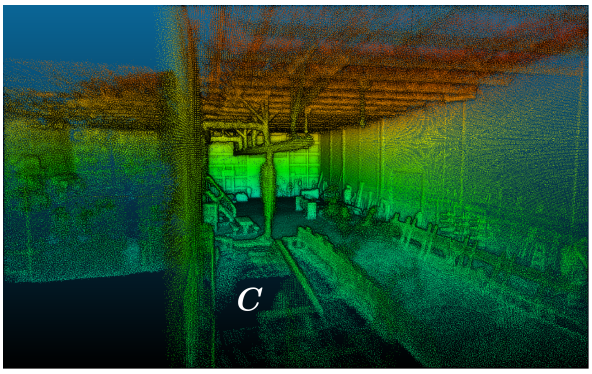
As shown in Chapter 4, the system has been integrated into real-world test flights. Following these flights, the drone pilot and SLAM developer emphasized that the scan was much more detailed and complete than in the remotely operated flights. In these flights, several shadow areas were apparent, even for areas within sight of the takeoff point, despite

3. As mentioned in the opening note in Chapter 3, the run time of RRT-Rope has been improved, with respect to the comparative results presented in the paper, as a result of internal system changes.

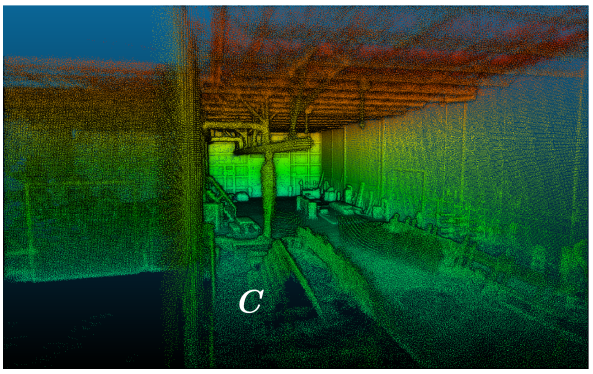
the pilot's commitment to producing a complete scan. Fig. 5.8 provides a comparison of scans after a real-world flight in Env. D. Some shadow areas from the manual flight are identified by A, B and C. Other comments included the advantage of not having to pilot and therefore monitor operations in a more relaxed manner. Also, the mechanical designer was comfortable with the constraints imposed on the system during autonomous maneuvers, thanks to the addition of path smoothing.



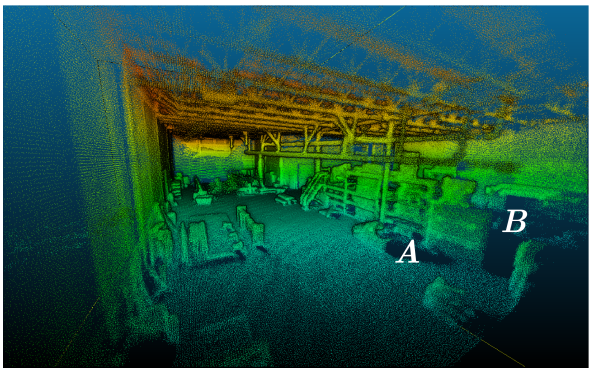
(a) Autonomous flight with onboard pictures.



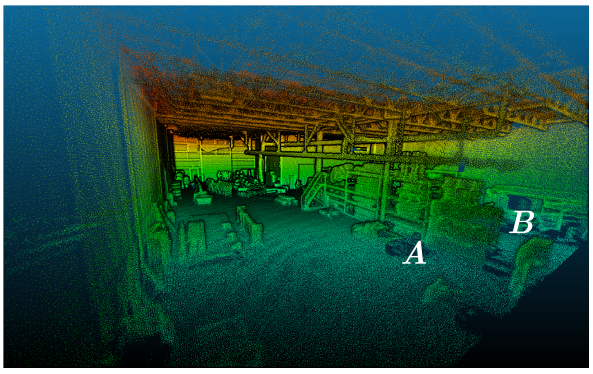
(b) Manual flight.



(c) Autonomous flight.



(d) Manual flight.



(e) Autonomous flight.

Figure 5.8 Comparison of scans after an autonomous and a manual exploration flight in a warehouse.

CHAPTER 6

DISCUSSION AND PERSPECTIVES

This chapter takes a step back from the work performed by analyzing the weaknesses of the system and discusses potential improvements. The algorithms presented in this thesis have been optimized to occupy the sweet spot between computation time and accuracy for the given problem. This chapter discusses ways to improve one factor, i.e., accuracy or computation time, at the cost of the other factor. The next sections deal with the aspects of path planning, exploration and tether modeling and management.

6.1 Path planning

The two main possibilities of extension of RRT-Rope concern the real-time calculation for a maze-like environment, and the possibility to include kinodynamic constraints for a more aggressive drone.

6.1.1 Online optimization

As mentioned in Section 3.6, RRT-Rope can be used online to quickly converge before continuing the search with Informed RRT*—with a smaller upper bound thanks to the path found with the shortcut—in order to escape local minima, and then start the procedure again. This idea was partly incorporated into an algorithm called F-RRT* [Liao et al., 2021]. However, this method differs from our suggested approach because it generates a parent node near obstacles, similarly to [Islam et al., 2012]. This algorithm shows good results for online optimization but an RRT-Rope shortcut—with the use of intermediate nodes while omitting ϵ—should give better results, as demonstrated in Chapter 3.

6.1.2 Kinodynamic constraints

The efficiency of RRT-Rope comes mainly from the knowledge of the shape of the solution for a holonomic problem in an uncluttered environment. The algorithm is therefore not applicable to problems with kinodynamic constraints, i.e., kinematic and/or dynamic. To take into account the kinematic constraints, a possible modification consists in integrating a steering function (e.g., Dubins path or Reeds-Shepp curves [Reeds and Shepp, 1990]) in the function "extend" of RRT and in the function "collisionFree" of RRT-Rope. The major disadvantage with this technique is that the RRT can no longer be constructed as quickly by omitting the infinitesimal distance ϵ because of the steering function. The advantage of RRT-Rope is thus weakened. Furthermore, for a drone whose dynamics is

important, this solution is not realistic since it assumes an instantaneous inclination to enter a turn. Techniques for kinodynamically constrained shortcutting exist, and they incorporate a control law [Hu et al., 2021]. Considering the dynamics, the heuristic based on the triangular inequality provides the shortest path but this path is not necessarily the most optimal, e.g., in terms of travel time. This makes the shortcutting problem complicated. However, for real-time integration, it is interesting to separate path planning and control into a hierarchical architecture. It allows better real-time execution, with a) one module responsible for finding a large-scale roughly feasible path without collisions, and b) the other module responsible for reacting to perturbations and following the desired trajectory as best as possible, within the constraints of the system.

6.2 Exploration

The avenues of improvement for exploration mainly concern the representation of frontier clusters and the extension of the algorithm for large maze-like environments. Other areas of improvement concern the tether model and management, which is discussed in Section 6.3.

6.2.1 Frontiers visibility

The approached presented in Section 4.4.2 for viewpoint generation is illustrated in Fig. 6.1. As discussed in the same section, a visibility-based approach can be integrated if computation time is not a limitation.

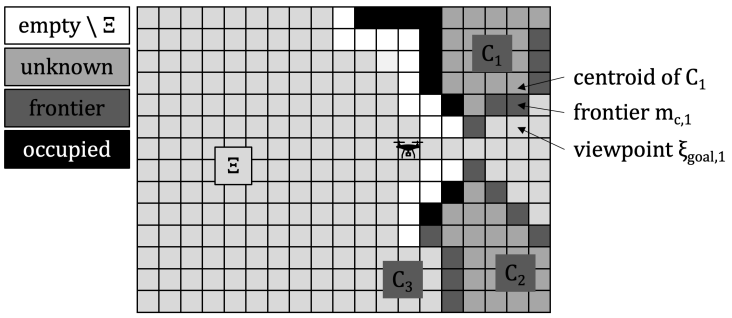


Figure 6.1 Frontier viewpoint generation.

Art Gallery Problem The naive way is to solve the *Art Gallery Problem* (AGP) for each frontier cluster. This visibility problem consists in finding a minimum number of fixed points (cameras) that allow to fully observe a defined space (an art gallery). Fig. 6.2 gives an overview of the problem with 2 cameras. This problem is not trivial in 3D, even for a polyhedron. More details are presented in [O'Rourke, 1987]. The first additional challenge for the present problem is that the solving must be performed by ray tracing

and cannot use geometric properties of a polyhedron. Also, in the classical problem, an infinite sensor range is assumed but the LiDAR finite range must be taken into account. Another problem-specific challenge is that some frontier voxels are not visible from the reachable space and these must be filtered accordingly. In order to group some clusters by the same point of view, [Kulich et al., 2019] solves the AGP before the frontier clustering, in 2D. This allows to reduce the size of the TSP to be solved afterwards.

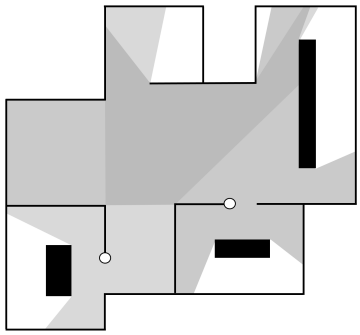


Figure 6.2 AGP representation for 2 cameras, represented by the white circles. The areas covered by one camera are in light gray and the overlap of the 2 cameras in dark gray.

Watchman Route Problem The *Watchman Route Problem* (WRP) is a derivative of the AGP that consists in finding the shortest path that a mobile watchman must take to guard an area whose map is known. The WRP thus unifies the AGP (which combines clustering and frontier representation) with the TSP. The problem is represented in Fig. 6.3. The ultimate approach for the problem studied in this work is to create a derivative of the WRP that also takes into account the tether unwinding process. The existing literature on the subject is very limited to date and focuses mainly on the case of known 2D polygons, whereas the problem studied here concerns an initially unknown 3D map [Mitchell, 2013, Mikula and Kulich, 2022]. This approach is therefore not conceivable for real-time computing with the current techniques and computing power.

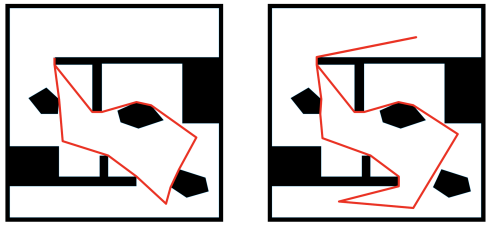
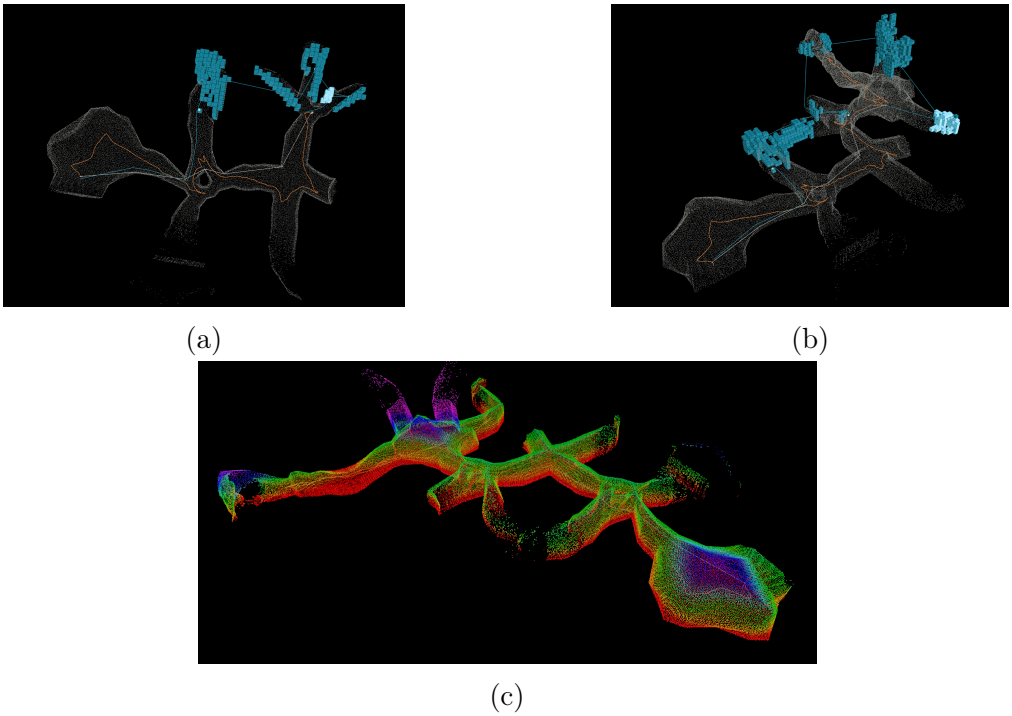


Figure 6.3 WRP solution in red, with an infinite sensor range (left) and finite sensor range (right) [Mikula and Kulich, 2022].

6.2.2 Large maze environments

While the previous subsection deals with the search for an optimal super-algorithm at the cost of computation time, this subsection focuses on the opposite, i.e., whether real-time integration is still feasible for a complicated environment significantly larger than the stopes, as targeted in Search And Rescue missions (e.g., an abandoned nuclear power plant, a stricken city, or a collapsed mine). As mentioned in Section 5.3, the limiting processes in terms of computation time for a growing problem are the path search for the adjacency matrix construction, and the tether model computation for a growing path history. Fig. 6.4 provides an example of a mission where the real-time computing condition (4 seconds) was not met, after 17 minutes of exploration.



(a) (b)

(c)

Figure 6.4 Exploration test in a simulated underground mining network: scan (a) after 7 minutes, (b) after 15 minutes, (c) after 17 minutes (with a Z-axis color scale). The tether model is in gray, the past trajectory in brown, and the computed exploration path in blue.

Table 6.1 gives exploration and computation time metrics at the end of this mission. The computation times are given for the last instance, which did not respect the real-time computing condition. Indeed, the computation time of TAPE is 7.5 s, above the specified limit of 4 s. Regarding the extra mission time, a trivial calculation for the distance traveled, at a speed of 0.5 m/s without hovering, gives a hypothetical mission time of 14.5 minutes. The overload of the computation therefore added about 2 minutes to the mission time. Regarding tether management, the final tether is less than the RTH cost, which indicates

that it was well managed. Fig. 6.4a confirms that the trajectory of the drone prevented the tether from getting stuck around the central pillar.

Table 6.1 Exploration metrics in an underground mining network.

Covered distance [m]	434.8
Mission duration [min]	17
Max. tether length [m]	136.6
Final tether length [m]	55.1
RTH cost [m]	56.3
Final TAPE computation time [s]	7.5
Final RRT-Rope computation time [s]	0.048
Final local planning computation time [s]	1.9

As mentioned in Section 5.3, the limiting factors are the adjacency matrix construction and the tether model computation in the local planner. For a real-time deployment in this kind of environment, the first part of the solution is to use Euclidean distances instead of path costs computed between all viewpoints. This will reduce the adjacency matrix construction time which contributes greatly to the total TAPE computation time. Then, to reduce the local planning time, the second part of the solution is to use a simpler model of the tether. This is developed in Section 6.3.

6.2.3 Insufficient tether length

The research was based on the assumption that the length of embedded tether is sufficient to get to the end of the stope by the shortest possible route. An interesting perspective of future work is to consider the case where the length of the embedded tether is insufficient to explore the whole stope. In this case, the algorithm must maximize the mapped surface or volume while respecting the tether constraint. One idea to develop is to integrate the tether computation in the TSP with a randomized improvement approach like SA, and to allow a number of visited cities lower than n. The limiting factor in this approach is the computation time with the current tether model and the complexity of the TSP, which would be increased by considering a variable number of visited cities.

6.3 Tether management

As said above, the tether calculation is computationally expensive and is an important area for improvement. A first approach of a simplified model is proposed to limit the computation time at the cost of accuracy. Then, the shortcomings of the current model are assessed, compared to reality.

6.3.1 Multi-resolution model

When the path history is growing, as presented in Fig. 6.4b, tracing the tether to find a possible relaxation of the problem with the first attachment points becomes demanding. A large part of the complexity of the model is finding the collision points in the OctoMap, which will determine the contact points. An improvement of this part consists in using the structure of the octree to compute a collision at low resolution (i.e., at low depth in the tree) and to refine the computation locally by going deeper in the tree in the case of a collision.

6.3.2 Multi-hypothesis belief

A limiting aspect of the tether model is its poor conditioning, i.e., a small change in the first points leads to a large change in the last points. Thus, if the first point of contact is a false positive (contact in simulation but not in reality), a false negative (the opposite), or poorly located due to localization or mapping errors, the model is likely to be far from reality at the end of the mission. The approach to address this problem is a multi-hypothesis belief system that model the uncertainties of the tether and keep track of the different probable configurations with an associated probability. Then, the approach takes into account the probability distribution in terms of tether length in the calculation of the cost function in TAPE. However, this strategy is costly in terms of computation time and memory allocation.

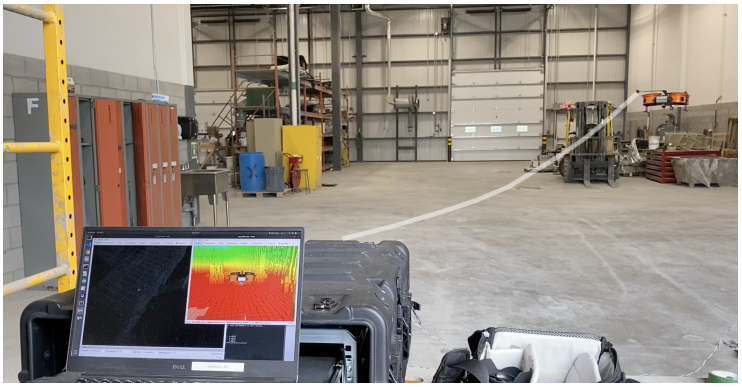


Figure 6.5 Highlight of the tether configuration during a flight.

6.3.3 NetherDrone design

As stated in Section 4.6, a shortcoming of the system is the lack of a rewind mechanism in the current design of the NetherDrone. Thanks to the global planning that solves the TSP, the average exploration path has few round trips and tests have shown that the tether often remains fairly taut for at least half the mission. Fig. 6.5 highlights the configuration of the tether during a flight. If the drone design were to remain unchanged, there are approaches

to bring the model closer to reality, up to a limit. However, a non-tensioned tether model with dynamic contact points does not currently exist to our knowledge, as mentioned in Section 2.2.4. Moreover, other variables such as the influence of the weight of the unwound tether complicate the problem, which then falls into the category of soft-body dynamics problems, whose solutions are known to be either inaccurate or not executable in real time. Some possible approaches include tracking the tether with a thermal camera or modeling a z-projection of the tether onto the ground. The envisaged limitations of the thermal camera are the visual obstruction of the tether by obstacles. The challenges of the ground projection model are the high number of contact points to calculate. Given the difficulty of a faithful real-time model, the most promising approach remains the integration of a rewind mechanism into the NetherDrone.

CHAPTER 7

CONCLUSION FRANÇAISE

Par rapport aux techniques actuelles de cartographie des chantiers miniers, l'utilisation d'un drone pallie les problèmes de zones d'ombres et de faible densité de points. Un drone filaire permet de partager la puissance de calcul avec une station de base, ainsi que garder la communication en temps réel avec un opérateur, et permettre une autonomie de vol illimitée. Lorsque le drone est téléopéré, plusieurs facteurs humains rendent la mission difficile. Plus particulièrement, le pilote éprouve des difficultés à percevoir un environnement en 3D à 360° ainsi qu'à comprendre cet environnement quand le drone navigue hors du champ de vision et que seule une représentation virtuelle est fournie. De plus, sans les aides visuelles, le pilote a une mauvaise idée de son point de décollage dans un environnement inconnu, ainsi que des zones qui n'ont pas été cartographiées quand l'environnement est large. Enfin, il est très difficile pour un pilote d'explorer différentes frontières tout en a) choisissant le chemin le plus court, b) en restant conscient du processus de déroulement du fil, et c) en minimisant la longueur du chemin global anticipé.

L'automatisation de la mission comble les lacunes liées à la téléopération humaine. Les travaux présentés dans cette thèse se concentrent sur le problème de navigation autonome d'un drone filaire destiné à la cartographie de chantiers miniers souterrains. Plus particulièrement, la recherche présentée développe des outils pour la planification de trajectoire et l'exploration autonome dans ces environnements. Les facteurs à prendre en compte sont la garantie de l'exhaustivité de la carte et la minimisation du temps de mission. Le temps de mission est étroitement lié à la distance à parcourir, et à la contrainte de calcul en temps réel, ce qui n'est pas trivial dans les environnements à grande échelle. L'exhaustivité de la carte (et donc de la mission d'exploration) nécessite une bonne gestion du fil. De plus, la longueur de fil embarqué contribue à la charge utile du drone et doit rester faible. Les techniques actuelles de planification et d'exploration des chemins ne sont pas suffisantes pour atteindre ces objectifs. Concernant la planification de trajectoire, le chapitre 2 montre que les arbres aléatoires sont très efficaces pour trouver un premier chemin, mais l'optimisation en ligne est coûteuse en temps de calcul et les techniques de raccourcissement nécessitent beaucoup d'itérations pour converger. De plus, ce nombre d'itérations est typiquement fixé par l'utilisateur de façon empirique, ce qui entraîne une performance variable et imprévisible. Concernant l'exploration, la techniques

des frontières est la plus représentative de l'espace à explorer et les techniques de sélection qui résolvent un *Traveling Salesman Problem* (TSP) permettent d'optimiser le chemin global. Enfin, les techniques d'exploration existantes qui considèrent un robot filaire ne s'attaquent qu'au cas 2D et n'optimisent pas le chemin global.

Pour relever les défis de la planification rapide de trajectoire dans des environnements à grande échelle, le chapitre 3 présente un nouvel algorithme, RRT-Rope, qui produit un chemin plus court ou égal aux algorithmes existants pour un problème holonome en un temps de calcul significativement plus court, jusqu'à 70% plus rapide que le deuxième meilleur algorithme dans un chantier minier représentatif. De plus, le comportement est plus régulier sans nécessiter un nombre d'itérations choisi par l'utilisateur. Les résultats montrent que la recherche d'un chemin est accélérée en supprimant le pas infinitésimal des arbres dans RRT-connect, ce qui réduit considérablement le nombre de générations de points nécessaires dans les grands environnements non encombrés. De plus, dans ces environnements, il est efficace de générer un premier chemin grossier mais faisable le plus rapidement possible, et de le raccourcir rapidement, menant ainsi à un chemin quasi-optimal grâce à la topologie de l'environnement. En effet, les résultats montrent que le chemin produit avec RRT-Rope est indépendant de la qualité du premier chemin dans ces environnements. De plus, les raccourcis inutiles sont évités grâce à une approche déterministe qui considère les points les plus éloignés en premier et détecte les lignes droites. Enfin, l'insertion de noeuds intermédiaires permet la quasi-optimalité et une résolution uniforme.

Pour relever les défis d'exploration avec un drone filaire, le chapitre 4 présente TAPE, la première méthode d'exploration de cavités en 3D qui se focalise sur la minimisation du temps de mission et la longueur du fil déroulé. La méthode emploie une architecture hiérarchique dont la planification globale résout un TSP pour des clusters de frontières donnés, et la planification locale minimise la distance parcourue tout en évitant les coincements du fil grâce à une fonction de décision dont les paramètres permettent de jouer sur le compromis entre ces deux métriques. En moyenne, TAPE entraine une augmentation de 4% de longueur de trajet global en plus que la méthode globale qui optimise le TSP, mais permet de garder le fil sous la longueur maximale autorisée dans 100% des cas, contre 53% avec la méthode initiale. En comparaison, le meilleur algorithme suivant en termes de longueur de chemin produit une augmentation de 25% de la distance par rapport à la méthode initiale. Les résultats montrent que les coincements du fil ne sont pas influencés par le chemin global mais fortement influencés par le chemin local, ce qui confirme la performance de l'approche hiérarchique. Grâce à cette approche, le planificateur local peut

être intégré avec n'importe quel autre planificateur global d'exploration. De plus, elle permet un calcul en temps réel plus facilement réalisable. Les résultats montrent également que la résolution du TSP de façon itérative, avec les dernières données de carte au cours de la mission, produit un chemin global quasi-optimal, malgré que la carte soit initialement inconnue.

Le chapitre 5 montre l'intégration de ces deux techniques en un outil d'exploration autonome pour drone filaire, qui produit une carte 3D complète des chantiers miniers en minimisant le temps de mission et en respectant le déroulement autorisé du fil. Les algorithmes de navigation autonome sont exécutés sur l'ordinateur de la station de base, et les commandes sont envoyées au contrôleur et à l'évitement d'obstacles embarqués qui réagissent à l'environnement en temps réel. Des fonctionnalités sont ajoutées au système pour mettre en œuvre les caractéristiques souhaitées par l'utilisateur. La trajectoire est ainsi exécutée et lissée pour répondre au problème réel qui est non-holonome. Une aide visuelle est fournie au pilote à travers l'affichage de zones à explorer dans une interface. Le pilote peut également sélectionner un point désiré dans l'espace reconstitué dans l'interface pour déclencher la navigation autonome vers ce point. Enfin, l'exploration autonome peut être déclenchée dans l'interface, avec des fonctionnalités qui permettent de limiter l'espace physique à explorer avec une boite de délimitation. Le temps de calcul est limité grâce à des astuces telles qu'une limitation du nombre de clusters, et une représentation multi-résolution et adaptative de l'espace. La mission autonome est assurée par des techniques de filtrages des frontières inatteignables et de modification de points en accord avec la carte dynamique. Le système a été largement testé dans le cadre de plus de 1000 missions d'exploration en simulation, et lors de plusieurs essais en vol dans le monde réel.

Suite aux limitations identifiées lors de ces tests, le chapitre 6 énonce les opportunités de travaux futurs incluant l'extension de la planification de trajectoire à un milieu encombré et l'ajout de contraintes cinématiques et dynamiques. Quant à l'exploration, une approche de visibilité des frontières intégrée dans un *Watchman Route Problem* (WRP) est considérée, et des pistes sont énoncées pour l'extension de l'algorithme à des grands environnements encombrés, ainsi que pour une longueur de fil insuffisante pour explorer entièrement l'espace. Enfin, des améliorations du modèle du fil sont proposées au travers d'un modèle multi-résolution pour accélérer le calcul, et d'un modèle qui considère les incertitudes pour une meilleure représentativité de la réalité. À long terme, toutes ces améliorations permettraient une navigation autonome plus générique dans tous les types d'environnements, y compris les zones industrielles, les villes, les réseaux miniers et les forêts.

CHAPTER 8

ENGLISH CONCLUSION

Compared to the current techniques for stope mapping, the use of a UAV overcomes the problems of shadow zones and low point density. A tethered drone enables computing power sharing with a base station, as well as communication in real time with an operator, and an unlimited flight time. When the drone is remotely operated, several human factors make the mission difficult. In particular, the pilot has difficulties to perceive a 3D environment at 360° as well as to understand this environment when the drone navigates out of the field of vision and only a virtual representation is provided. Moreover, without visual aids, the pilot has a poor idea of his take-off point in an unknown environment, as well as of the areas that have not been mapped when the environment is large. Finally, it is very difficult for a pilot to explore different frontiers while a) choosing the shortest path, b) maintaining awareness of the tether unwinding process, and c) minimizing the length of the anticipated global path.

The automation of the mission fills the gaps related to human teleoperation. The work presented in this thesis focuses on the autonomous navigation problem of a tethered drone intended for the mapping of underground mining stopes. More specifically, it develops tools for path planning and autonomous exploration in these environments. Factors to consider are ensuring the completeness of the map and minimizing the mission time. The mission time is closely related to the distance to be covered, and to the real-time computing constraint, which is not trivial in large-scale environments. The completeness of the map (and thus of the exploration mission) requires a good tether management. Also, the length of the onboard tether contributes to the drone payload and should be kept small. Current path planning and exploration techniques are not sufficient to meet these objectives. Regarding path planning, Chapter 2 shows that random trees are very effective for finding a first path, but online optimization is computationally expensive, and shortening techniques require many iterations to converge. Moreover, this number of iterations is typically set empirically by the user, resulting in variable and unpredictable performance. Regarding exploration, the frontier techniques are the most representative of the space to be explored and selection techniques that solve a Traveling Salesman Problem (TSP) ensure the optimization of the global path. Finally, existing exploration techniques

that consider a tethered robot only address the 2D case and do not optimize the global path.

To address the challenges of fast path planning in large-scale environments, Chapter 3 presents a new algorithm, RRT-Rope, which produces a path shorter than or equal to existing algorithms for a holonomic problem in a significantly shorter computation time, up to 70% faster than the second best algorithm in a representative stope. In addition, the behavior is more consistent—in solution quality and computation time—without requiring a user-selected number of iterations. The results show that path finding is accelerated by removing the infinitesimal tree step in RRT-connect, which significantly reduces the number of point generations required in large-scale uncluttered environments. Moreover, in these environments, it is efficient to generate a first rough but feasible path as quickly as possible, and to shorten it quickly. This technique leads to a near-optimal path by leveraging the topology of the environment. Indeed, the results indicate that the path produced with RRT-Rope is independent of the quality of the first path in these environments. Moreover, unnecessary shortcuts are avoided thanks to a deterministic approach that considers the farthest points first and detects straight lines. Finally, the insertion of intermediate nodes results in quasi-optimality and uniform resolution.

To meet the challenges of exploration with a tethered drone, Chapter 4 presents TAPE, the first 3D cavity exploration method that focuses on minimizing mission time and unwound tether length. The method employs a hierarchical architecture in which the global planner solves a TSP for given frontier clusters, and the local planner minimizes the distance traveled while avoiding tether entanglements thanks to a decision function whose parameters enable to play on the trade-off between these two metrics. On average, TAPE leads to a 4% increase in global path length compared to the global method that optimizes the TSP, but keeps the tether below the maximum allowed length in 100% of the cases, compared to 53% with the initial method. In comparison, the next best algorithm in terms of path length produces a 25% increase in distance compared to the initial method. The results revealed that the tether entanglements are not influenced by the global path but strongly influenced by the local path, which confirms the performance of the hierarchical approach. With this approach, the local planner can be integrated with any other global exploration planner. In addition, it leads to a more easily achievable real-time computation. The results also indicate that solving the TSP iteratively, with the latest map data during the mission, produces a near-optimal global path, despite the map being initially unknown.

Chapter 5 presents the integration of these two new algorithms into an autonomous tethered drone exploration tool, which produces a complete 3D map of the stopes while min-

imizing mission time and respecting the authorized tether unwinding. Autonomous navigation algorithms are executed on the base station computer, and commands are sent to the onboard local obstacle avoidance system that reacts to the environment in real time. Functionalities are added to the system to implement the features desired by the user. The trajectory is thus executed and smoothed to satisfy the real problem which is non-holonomic. A visual aid is provided to the pilot through the display of areas to explore in an interface. The pilot can also select a desired point in the reconstructed space in the interface to trigger autonomous navigation to that point. Finally, the autonomous exploration can be triggered in the interface, with features that let the user to limit the physical space to explore with a bounding box. The computation time is limited thanks to tricks such as a limitation of the number of clusters, and a multi-resolution and adaptive representation of the space. Autonomous mission is ensured by techniques for filtering out unreachable frontiers and modifying points in accordance with the dynamic map. The system has been extensively tested in more than 1,000 exploration missions in simulation, and several real-world flight tests.

Following the limitations identified during these tests, Chapter 6 outlines opportunities for future work including extending path planning to a cluttered environment and adding kinematic and dynamic constraints. As for the exploration, a frontier visibility approach integrated into a Watchman Route Problem (WRP) is considered, and avenues are stated for extending the algorithm to large maze environments, as well as to a tether length insufficient to fully explore the environment. Finally, improvements of the tether model are proposed through a multi-resolution model to speed up the computation, and a model that considers uncertainties for a better representation of reality. In the long run, all these improvements would allow a more generic autonomous navigation in all types of environments, including industrial areas, cities, mining networks and forests.

APPENDIX A
Real-time requirement

For localization, mapping, and control purposes, the NetherDrone moves at a near-constant speed of 0.5 meters per second. This allows to generate a preliminary estimate of the mission time, in order to set the objectives in terms of calculation time. For a narrow stope of 150 meters in height, it corresponds, at best, to a travel time of 10 minutes to observe the shadow zones at the bottom of the stope.

During this time, the path must be regenerated several times according to the map updates, provided at each full rotation of the LiDAR every 4 seconds. This corresponds to more than 150 instances. These instances are composed of several path requests for 2 reasons: a) an optimization of the global path distance, in order not to lengthen the mission time, b) a local optimization of the path to avoid tether entanglements, in order not to unroll tether unnecessarily and risk not being able to finish the mission due to lack of tether length.

A reasonable maximum number of waypoints for the optimization of the global path is 15, to avoid overloading the calculation but to keep a good representation of the remaining space to map. Knowing that the number of edges in a complete undirected graph is $n\,(n-1)/2$, about 105 path requests are necessary just to know the costs of the paths between the waypoints, and 20 path requests are required, on average, for local path optimization.

In total, 125 path requests are necessary at each instance, i.e., 19,000 during the mission. This number does not include additional calculations consisting of, among others, replanning if the control of the drone failed to execute the desired trajectory in a sufficiently precise way, optimization of the global path among the possible paths (see Section 2.3), and data pre and post-processing (see Chapters 3 and 4). Taking into account only the time of path requests, a naive solution with an arbitrary time of 0.1 second per path request would lead to a total computation time of more than 30 minutes for the whole mission, which represents 3 times the expected mission time. This quick estimate gives an idea of the total mission delay if the system is not able to run in real time.

APPENDIX B

Concorde TSP Solver

The Concorde TSP Solver [Applegate et al., 2006] holds the current record for the largest solved problem (85,900 cities in 136+ CPU years) with an exact algorithm. It uses a branch-and-cut method, which is a combination of BnB and the cutting-plane method [Gilmore and Gomory, 1961]. This latter, used in Linear Programming (LP), allows to relax the problem by temporarily violating the TSP cycle constraint.

B.1 Algorithm

The temporarily produced cycle is called a fractional cycle. By defining a variable x_{ij}, which determines whether the branch between i and j is part of the tour, the traveler can use non-binary fractions of branches so that $0 \leq x_{ij} \leq 1$. For example, to get to and from vertex 2, a suitable value set is $\{x_{12} = 0.5, x_{23} = 0.5, x_{24} = 1\}$, with the constraint that the sum of the variables adjacent to each vertex must be 2, i.e.,

$$\Sigma_j x_{2j} = 2.$$

If the branch cost is noted as c_{ij}, the TSP becomes an optimization problem which aims to minimize the function

$$f = \Sigma_{i,j} c_{ij} x_{ij}.$$

The solution is then refined by correcting the flaws produced (i.e., subtours) through adding corresponding constraints. An example of a constraint is, for a subtour of 2 points containing a round trip, to impose that the edge variables leaving and arriving at the union of these 2 points must be equal to 2. The x_{ij} eventually converge to integers and optimality is met through the dual LP problem.

B.2 Optimality

The dual problem of a LP is a linear problem that is derived from the primal LP. Each variable in the primal LP is expressed as a constraint in the dual LP and each constraint in the primal LP is expressed as a variable in the dual LP. In the Concorde TSP Solver, the variables from the primal LP (x_{ij}) lead to inequalities (constraints) to remove some subtours. The dual problem is performed through a zone packing problem of control zones, represented as disks of radius r_i centered on each vertex. In the dual problem, these zones cannot overlap, i.e.,

$$r_i + r_j \leq c_{ij}.$$

Since a proper tour must cross the control areas at least twice, any tour will be larger than $2\Sigma_i r_i$. The zone packing problem maximizes the function

$$f = \Sigma_i r_i,$$

which gives a lower bound to the solution of the primal problem, according to the weak duality theorem. If the control areas geometrically cover all the edges, and the fractional tour is optimal, then the strong duality theorem states that the length of the minimal tour is equal to the bound given by the dual LP. For the TSP, any legal tour will be equal or larger than this value. So, if the fractional tour is legal, it means that the optimum is reached. More details are presented in [Applegate et al., 2006].

APPENDIX C

Christofides algorithm

Approximation algorithms, unlike heuristics, promise a worst-case performance and are part of a more formal framework. The most popular approximation algorithm promises an approximation factor of 1.5 [Christofides, 1976].

C.1 Algorithm

Christofides first creates a Minimum Spanning Tree (MST) [1] from the graph. Then, a minimum-weight perfect matching [2] M is found between the odd degree vertices. By combining MST and M in a graph, a Eulerian cycle [3] is searched. Finally, the cities that appear several times in the Eulerian cycle are skipped by shortcutting to produce a Hamiltonian cycle. The proof of the approximation ratio is presented below. More details are given in [Christofides, 1976].

C.2 Approximation ratio

Considering a tour T_{MST} constructed with a DFS in the MST, each edge must be visited exactly 2 times, so that

$$c(T_{MST}) = 2\,c(MST).$$

By removing a random edge e in the optimal tour of the TSP, denoted T^*, the tree $\{T^* - e\}$ is obtained. The cost $c(MST)$ is, by definition, less than the cost of this tree. Therefore, the approximation factor of T_{MST} is 2. Considering now the cost of the Christofides tour,

$$c(T_{\text{Christofides}}) = c(MST) + c(M).$$

As mentioned above,

$$c(MST) \leq c(T^*).$$

If the tour T^* is divided into 2 complementary perfect matchings M_1 and M_2, it is obvious that the cost of the minimum-weight perfect matching $c(M)$ is less than $c(M_1)$ and $c(M_2)$. Since shortcutting does not increase the cost in a metric TSP thanks to triangular inequality, this translates into

$$c(T_{\text{Christofides}}) \leq c(T^*) + \frac{1}{2}\,c(T^*),$$

1. A spanning tree is a tree that include all vertices in a graph. A MST is a spanning tree whose sum of edge costs is minimal.

2. A matching is a set of edges without common vertices. A perfect matching is a matching that covers every vertex.

3. A cycle that visits every edge exactly one, allowing for revisiting vertices.

i.e., the approximation factor of 1.5. As mentioned earlier, this upper bound is not guaranteed by heuristics but some give much better performances than the approximation algorithms.

APPENDIX D

Lin-Kernighan heuristic pseudo-code

Algorithm 7: Lin-Kernighan pseudo-code

1 Generate a random initial tour T.
2 $i \leftarrow 1$;
3 Choose a random city t_1.
4 Choose $x_1 \leftarrow (t_1, t_2) \in T$;
5 Choose $y_1 \leftarrow (t_2, t_3) \notin T : G_1 > 0$;
6 **if** such y_1 does not exist **then**
7 Go to step 31.
8 $i \leftarrow i + 1$;
9 Choose $x_i \leftarrow (t_{2i-1}, t_{2i}) \in T : x_i \neq y_s, \forall s < i$;
10 $T' \leftarrow$ resulting configuration if t_{2i} is joined to t_i.
11 **if not** isTour(T') **then**
12 Go to step 22.
13 **if** $c(T') < c(T)$ **then**
14 $T \leftarrow T'$;
15 Go to Step 2.
16 Choose $y_i \leftarrow (t_{2i}, t_{2i+1}) \notin T : G_i > 0$ **and** $y_i \neq x_s, \forall s \leq i$ **and** x_{i+1} exists.
17 **if** such y_i exists **then**
18 Go to step 8.
19 **if** there is an untried alternative for y_2 **then**
20 $i \leftarrow 2$;
21 Go to step 16.
22 **if** there is an untried alternative for x_2 **then**
23 $i \leftarrow 2$;
24 Go to step 9.
25 **if** there is an untried alternative for y_1 **then**
26 $i \leftarrow 1$;
27 Go to step 5.
28 **if** there is an untried alternative for x_1 **then**
29 $i \leftarrow 1$;
30 Go to step 4.
31 **if** there is an untried alternative for t_1 **then**
32 Go to step 2.
33 **return** T;

LIST OF REFERENCES

[Alhamdy et al., 2012] Alhamdy, S., Noudehi, A., and Majdara, M. (2012). Solving traveling salesman problem using ants colony algorithm and comparing with tabu search, simulated annealing and genetic algorithm. *Journal of Applied Sciences Research*, 8:434–440.

[Applegate et al., 2003] Applegate, D., Cook, W. J., and Rohe, A. (2003). Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15:82–92.

[Applegate et al., 2006] Applegate, D. L., Bixby, R. E., Chvatál, V., and Cook, W. J. (2006). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

[Arslan and Tsiotras, 2013] Arslan, O. and Tsiotras, P. (2013). Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *IEEE International Conference on Robotics and Automation*, pages 2421–2428.

[Atlas copco, 2022] Atlas copco (2022). Mining methods in underground mining. `https://miningandblasting.wordpress.com`. [Online; accessed 06-September-2022].

[Bachrach et al., 2010] Bachrach, A., de Winter, A., He, R., Hemann, G., Prentice, S., and Roy, N. (2010). Range - robust autonomous navigation in gps-denied environments. In *IEEE International Conference on Robotics and Automation*, pages 1096–1097.

[Bai et al., 2016] Bai, S., Wang, J., Chen, F., and Englot, B. (2016). Information-theoretic exploration with bayesian optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1816–1822.

[Bailey et al., 2000] Bailey, C., McLain, T., and Beard, R. (2000). Fuel saving strategies for separated spacecraft interferometry. In *AIAA Guidance, Navigation and Control Conference*.

[Balta et al., 2016] Balta, H., Bedkowski, J., Govindaraj, S., Majek, K., Musialik, P., Serrano, D., Alexis, K., Siegwart, R., and De Cubber, G. (2016). Integrated data management for a fleet of search-and-rescue robots. *Journal of Field Robotics*, 34:539–582.

[Bar-Noy and Schieber, 1991] Bar-Noy, A. and Schieber, B. (1991). The canadian traveller problem. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 261–270.

[Bellman, 1962] Bellman, R. (1962). Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9(1):61–63.

[Berchtold and Glavina, 1994] Berchtold, S. and Glavina, B. (1994). A scalable optimizer for automatically generated manipulator motions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1796–1802.

[Bhattacharya et al., 2010] Bhattacharya, S., Kumar, V., and Likhachev, M. (2010). Search-based path planning with homotopy class constraints. In *AAAI Conference on Artificial Intelligence*, page 1230–1237.

[Bhattacharya et al., 2012] Bhattacharya, S., Likhachev, M., and Kumar, V. (2012). Topological constraints in search-based robot path planning. *Autonomous Robots*, 33.

[Bi et al., 2021] Bi, H., Yang, Z., and Wang, M. (2021). The performance of different algorithms to solve traveling salesman problem. In *International Conference on Big Data & Artificial Intelligence & Software Engineering*, pages 153–156.

[Biggs, 1981] Biggs, N. L. (1981). T. p. kirkman, mathematician. *Bulletin of the London Mathematical Society*, 13(2):97–120.

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

[Bircher et al., 2016] Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding horizon "next-best-view" planner for 3d exploration. In *IEEE International Conference on Robotics and Automation*, pages 1462–1468.

[Brock and Khatib, 2002] Brock, O. and Khatib, O. (2002). Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.

[Cao et al., 2021a] Cao, C., Zhu, H., Choset, H., and Zhang, J. (2021a). Exploring large and complex environments fast and efficiently. In *IEEE International Conference on Robotics and Automation*, pages 7781–7787.

[Cao et al., 2021b] Cao, C., Zhu, H., Choset, H., and Zhang, J. (2021b). Tare: A hierarchical framework for efficiently exploring complex 3d environments. *Robotics: Science and Systems XVII*.

[Caserta and Voß, 2014] Caserta, M. and Voß, S. (2014). A hybrid algorithm for the dna sequencing problem. *Discrete Applied Mathematics*, 163:87–99.

[Castro et al., 2008] Castro, R., Lewiner, T., Lopes, H., Tavares, G., and Bordignon, A. (2008). Statistical optimization of octree searches. *Computer Graphics Forum*.

[Christofides, 1976] Christofides, N. (1976). Worst-case analysis of a new heuristic for the traveling salesman problem. *Carnegie Mellon University*, 3:10.

[Cook, 2011] Cook, W. J. (2011). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.

[Cosares et al., 1995] Cosares, S., Deutsch, D., Saniee, I., and Wasem, O. (1995). Sonet toolkit: A decision support system for designing robust and cost-effective fiber-optic networks. *Interfaces*, 25:20–40.

[Dang et al., 2019] Dang, T., Mascarich, F., Khattak, S., Papachristos, C., and Alexis, K. (2019). Graph-based path planning for autonomous robotic exploration in subterranean environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3105–3112.

[Dang et al., 2018] Dang, T., Papachristos, C., and Alexis, K. (2018). Autonomous exploration and simultaneous object search using aerial robots. In *IEEE Aerospace Conference*, pages 1–7.

[Dang et al., 2020] Dang, T., Tranzatto, M., Khattak, S., Mascarich, F., Alexis, K., and Hutter, M. (2020). Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*.

[Dantzig and Ramser, 1959] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science (pre-1986)*, 6(1):80.

[Delorme et al., 2019] Delorme, X., Dolgui, A., Kovalev, S., and Kovalyov, M. Y. (2019). Minimizing the number of workers in a paced mixed-model assembly line. *European Journal of Operational Research*, 272(1):188–194.

[Dijkstra, 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

[Dorigo and Gambardella, 1997] Dorigo, M. and Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73–81.

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

[Dornhege and Kleiner, 2011] Dornhege, C. and Kleiner, A. (2011). A frontier-void-based approach for autonomous exploration in 3d. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 351–356.

[Dozois, 2022] Dozois, D. (2022). *Développement de technologies utilisant un algorithme de cartographie et localisation simultanées et utilisant un algorithme de champ de potentiel permettant l'évitement d'obstacles lors du pilotage d'un drone dans un chantier minier*. Master's thesis, Université de Sherbrooke, Sherbrooke, Canada.

[Dry et al., 2006] Dry, M., Lee, M., Vickers, D., and Hughes, P. (2006). Human performance on visually presented traveling salesperson problems with varying numbers of nodes. *The Journal of Problem Solving*, 1.

[EarthSense, 2022] EarthSense (2022). Levels of autonomy for field robots. `https://www.earthsense.co/news/2020/7/24/levels-of-autonomy-for-field-robots`. [Online; accessed 8-September-2022].

[Emesent, 2019] Emesent (2019). Hovermap. `https://emesent.io/products`. [Online; accessed 5-October-2019].

[Fox and Burgard, 1999] Fox, D. and Burgard, W. (1999). Markov localization for mobile robots in dynamic environments. *Journal Of Artificial Intelligence Research*, 11:391–427.

[Freda et al., 2008] Freda, L., Oriolo, G., and Vecchioli, F. (2008). Sensor-based exploration for general robotic systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2157–2164.

[Freisleben and Merz, 1996] Freisleben, B. and Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *IEEE International Conference on Evolutionary Computation*, pages 616–621.

[Gammell et al., 2018] Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S. (2018). Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics*, 34(4):966–984.

[Gammell et al., 2014] Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014). Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004.

[Gao et al., 2018] Gao, W., Booker, M., Adiwahono, A., Yuan, M., Wang, J., and Yun, Y. W. (2018). An improved frontier-based approach for autonomous exploration. In *International Conference on Control, Automation, Robotics and Vision*, pages 292–297.

[Garey and Johnson, 1979] Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. Freeman.

[Genova and Williamson, 2017] Genova, K. and Williamson, D. P. (2017). An experimental evaluation of the best-of-many christofides' algorithm for the traveling salesman problem. *Algorithmica*, 78:1109–1130.

[Geraerts and Overmars, 2004] Geraerts, R. and Overmars, M. H. (2004). Clearance based path optimization for motion planning. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2386–2392.

[Gilmore and Gomory, 1961] Gilmore, R. and Gomory, R. (1961). A linear programming approach to the cutting stock problem. *Operations Research*, 9.

[González-Baños and Latombe, 2002] González-Baños, H. H. and Latombe, J.-C. (2002). Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848.

[Grocholsky et al., 2006] Grocholsky, B., Keller, J., Kumar, V., and Pappas, G. (2006). Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13(3):16–25.

[Gutin et al., 2002] Gutin, G., Yeo, A., and Zverovich, A. (2002). Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp. *Discrete Applied Mathematics*, 117(1):81–86.

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

[Hauser, 2022] Hauser, K. (2022). *Robotic Systems*, chapter 3. University of Illinois at Urbana-Champaign.

[Held and Karp, 1962] Held, M. and Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210.

[Helsgaun, 2000] Helsgaun, K. (2000). An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130.

[Holz et al., 2010] Holz, D., Basilico, N., Amigoni, F., and Behnke, S. (2010). Evaluating the efficiency of frontier-based exploration strategies. In *International Symposium on Robotics and German Conference on Robotics*, pages 1–8.

[Hong et al., 2019] Hong, A., Igharoro, O., Liu, Y., Niroui, F., Nejat, G., and Benhabib, B. (2019). Investigating human-robot teams for learning-based semi-autonomous control in urban search and rescue environments. *Journal of Intelligent and Robotic Systems*.

[Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*.

[Hu et al., 2021] Hu, B., Cao, Z., and Zhou, M. (2021). An efficient rrt-based framework for planning short and smooth wheeled robot motion under kinodynamic constraints. *IEEE Transactions on Industrial Electronics*, 68:3292–3302.

[Hu and Raidl, 2008] Hu, B. and Raidl, G. (2008). Effective neighborhood structures for the generalized traveling salesman problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 36–47.

[Islam et al., 2012] Islam, F., Nasir, J., Malik, U., Ayaz, Y., and Hasan, O. (2012). Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution. In *IEEE International Conference on Mechatronics and Automation*, pages 1651–1656.

[Ito et al., 1996] Ito, N., Kamekura, R., Shimazu, Y., Yokoyama, T., and Matsushita, Y. (1996). The combination of edge detection and region extraction in nonparametric color image segmentation. *Information Sciences*, 92(1):277–294.

[Johnson and McGeoch, 2018] Johnson, D. S. and McGeoch, L. A. (2018). *The traveling salesman problem: a case study*, pages 215–310. Princeton University Press, Princeton.

[Juliá et al., 2012] Juliá, M., Gil, A., and Reinoso, Ó. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33:427–444.

[Kam et al., 2015] Kam, H. R., Lee, S.-H., Park, T., and Kim, C.-H. (2015). Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60:337–345.

[Kanehara et al., 2007] Kanehara, M., Kagami, S., Kuffner, J., Thompson, S., and Mizoguhi, H. (2007). Path shortening and smoothing of grid-based path planning with consideration of obstacles. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 991–996.

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

[Karaman et al., 2011] Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime motion planning using the rrt*. In *IEEE International Conference on Robotics and Automation*, pages 1478–1483.

[Keidar and Kaminka, 2014] Keidar, M. and Kaminka, G. A. (2014). Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, 33(2):215–236.

[Kernighan and Lin, 1970] Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307.

[Khan and Asadujjaman, 2016] Khan, M. M.-U.-R. and Asadujjaman, M. (2016). A tabu search approximation for finding the shortest distance using traveling salesman problem. *IOSR Journal of Mathematics*, 12:80–84.

[Kim et al., 2013] Kim, S., Bhattacharya, S., Ghrist, R., and Kumar, V. (2013). Topological exploration of unknown and partially known environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3851–3858.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

[Koenig and Howard, 2004] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3.

[Koren and Borenstein, 1991] Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 1398–1404 vol.2.

[Kuffner and LaValle, 2000] Kuffner, J. and LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 995–1001.

[Kulich et al., 2011] Kulich, M., Faigl, J., and Přeučil, L. (2011). On distance utility in the exploration task. In *IEEE International Conference on Robotics and Automation*, pages 4455–4460.

[Kulich et al., 2019] Kulich, M., Faigl, J., and Přeučil, L. (2019). An integrated approach to goal selection in mobile robot exploration. *Sensors*, 19(6):1400.

[Kumar and Luo, 2003] Kumar, R. and Luo, Z. (2003). Optimizing the operation sequence of a chip placement machine using tsp model. *IEEE Transactions on Electronics Packaging Manufacturing*, 26(1):14–21.

[Labbé and Michaud, 2018] Labbé, M. and Michaud, F. (2018). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36.

[LaValle, 2006] LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.

[LaValle and Kuffner, 2000] LaValle, S. M. and Kuffner, J. (2000). Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics*, pages 293–308.

[Liao et al., 2021] Liao, B., Wan, F., Hua, Y., Ma, R., Zhu, S., and Qing, X. (2021). F-rrt*: An improved path planning algorithm with improved initial solution and convergence rate. *Expert Systems with Applications*, 184:115457.

[Lin and Kernighan, 1973] Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516.

[Lindemann and LaValle, 2001] Lindemann, S. R. and LaValle, S. M. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400.

[Lluvia et al., 2021] Lluvia, I., Lazkano, E., and Ansuategi, A. (2021). Active mapping and robot exploration: A survey. *Sensors*, 21(7).

[Luna et al., 2013] Luna, R., Şucan, I. A., Moll, M., and Kavraki, L. E. (2013). Anytime solution optimization for sampling-based motion planning. In *IEEE International Conference on Robotics and Automation*, pages 5068–5074.

[Lundy and Mees, 1986] Lundy, M. and Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical Programming*, 34(1):111–124.

[Luo and Hauser, 2014] Luo, J. and Hauser, K. (2014). An empirical study of optimal motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1761–1768.

[Lynch and Park, 2017] Lynch, K. M. and Park, F. C. (2017). Motion planning. In *Modern robotics: mechanics, planning, and control*, pages 353–402. Cambridge University Press, 1 edition.

[Macgregor and Ormerod, 1996] Macgregor, J. and Ormerod, T. (1996). Human performance on the traveling salesman problem. *Perception & psychophysics*, 58:527–39.

[Martínez-Rozas et al., 2021] Martínez-Rozas, S., Alejo, D., Caballero, F., and Merino, L. (2021). Optimization-based trajectory planning for tethered aerial robots. In *IEEE International Conference on Robotics and Automation*, pages 362–368.

[Martínez-Rozas et al., 2022] Martínez-Rozas, S., Alejo, D., Caballero, F., and Merino, L. (2022). Path and trajectory planning of a tethered uav-ugv marsupial robotics system.

[Mashayekhi et al., 2020] Mashayekhi, R., Idris, M. Y. I., Anisi, M. H., Ahmedy, I., and Ali, I. (2020). Informed rrt*-connect: An asymptotically optimal single-query path planning method. *IEEE Access*, 8:19842–19852.

[McCammon and Hollinger, 2017] McCammon, S. and Hollinger, G. A. (2017). Planning and executing optimal non-entangling paths for tethered underwater vehicles. In *IEEE International Conference on Robotics and Automation*, pages 3040–3046.

[Meagher, 1980] Meagher, D. (1980). Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer. *Rensselaer Polytechnic Institute (Technical Report IPL-TR-80-111)*.

[Mikula and Kulich, 2022] Mikula, J. and Kulich, M. (2022). Towards a continuous solution of the d-visibility watchman route problem in a polygon with holes. *IEEE Robotics and Automation Letters*, 7(3):5934–5941.

[Mitchell, 2013] Mitchell, J. S. B. (2013). Approximating watchman routes. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 844–855.

[Montiel Ross and Delgadillo, 2015] Montiel Ross, O. and Delgadillo, F. (2015). Reducing the size of combinatorial optimization problems using the operator vaccine by fuzzy selector with adaptive heuristics. *Mathematical Problems in Engineering*, pages 1–14.

[Moravec and Elfes, 1985] Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121.

[Nehmzow, 2003] Nehmzow, U. (2003). Navigation. In *Mobile Robotics: A Practical Introduction*, pages 95–166. Springer London.

[O'Rourke, 1987] O'Rourke, J. (1987). *Art Gallery Theorems and Algorithms*. Oxford University Press, Inc., USA.

[Otte and Correll, 2013] Otte, M. and Correll, N. (2013). C-forest: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics*, 29(3):798–806.

[Papachristos et al., 2019] Papachristos, C., Khattak, S., Mascarich, F., and Alexis, K. (2019). Autonomous navigation and mapping in underground mines using aerial robots. In *IEEE Aerospace Conference*, pages 1–8.

[Pepper et al., 2002] Pepper, J., Golden, B., and Wasil, E. (2002). Solving the traveling salesman problem with annealing-based heuristics: a computational study. *IEEE Transactions on Systems, Man, and Cybernetics*, 32(1):72–77.

[Petit and Lussier Desbiens, 2021] Petit, L. and Lussier Desbiens, A. (2021). Rrt-rope: A deterministic shortening approach for fast near-optimal path planning in large-scale uncluttered 3d environments. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 1111–1118.

[Petit and Lussier Desbiens, 2022] Petit, L. and Lussier Desbiens, A. (2022). Tape: Tether-aware path planning for autonomous exploration of unknown 3d cavities using a tangle-compatible tethered aerial robot. *IEEE Robotics and Automation Letters*, 7(4):10550–10557.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3.

[Quin et al., 2014] Quin, P., Alempijevic, A., Paul, G., and Liu, D. (2014). Expanding wavefront frontier detection: An approach for efficiently detecting frontier cells. In *Australasian Conference on Robotics and Automation*.

[Quin et al., 2021] Quin, P., Nguyen, D., Vu, T., Alempijevic, A., and Paul, G. (2021). Approaches for efficiently detecting frontier cells in robotics exploration. *Frontiers in Robotics and AI*, 8:616470.

[Ravankar et al., 2018] Ravankar, A., Ravankar, A., Kobayashi, Y., and Emaru, T. (2018). Autonomous mapping and exploration with unmanned aerial vehicles using low cost sensors. In *International Electronic Conference on Sensors and Applications*.

[Reeds and Shepp, 1990] Reeds, J. and Shepp, L. (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393.

[Rusu, 2010] Rusu, R. B. (2010). Semantic 3d object maps for everyday manipulation in human living environments. *KI - Künstliche Intelligenz*, 24:345–348.

[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation*, pages 1–4.

[SAE, 2018] SAE (2018). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. In *SAE Standard J3016*, USA.

[Saffre et al., 2022] Saffre, F., Hildmann, H., Karvonen, H., and Lind, T. (2022). Monitoring and cordoning wildfires with an autonomous swarm of unmanned aerial vehicles. *Drones*, 6(10).

[Shade, 2011] Shade, R. (2011). *Choosing Where To Go: Mobile Robot Exploration*. PhD thesis, University of Oxford, Oxford, England.

[Shapovalov and Pereira, 2020a] Shapovalov, D. and Pereira, G. A. S. (2020a). Exploration of unknown environments with a tethered mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6826–6831.

[Shapovalov and Pereira, 2020b] Shapovalov, D. and Pereira, G. A. S. (2020b). Tangle-free exploration with a tethered mobile robot. *Remote Sensing*, 12(23).

[Shen et al., 2012] Shen, S., Michael, N., and Kumar, V. (2012). Autonomous indoor 3d exploration with a micro-aerial vehicle. In *IEEE International Conference on Robotics and Automation*.

[Shkolnik et al., 2009] Shkolnik, A., Walter, M., and Tedrake, R. (2009). Reachability-guided sampling for planning under differential constraints. In *IEEE International Conference on Robotics and Automation*, pages 2859 – 2865.

[Siegwart and Nourbakhsh, 2004] Siegwart, R. and Nourbakhsh, I. R. (2004). Planning and navigation. In *Introduction to Autonomous Mobile robot*, pages 257–304. MIT Press, 1 edition.

[Silver et al., 2006a] Silver, D., Carsten, J., and Thayer, S. (2006a). Topological global localization for subterranean voids. In *Field and Service Robotics*, pages 117–128. Springer Berlin Heidelberg.

[Silver et al., 2006b] Silver, D., Ferguson, D., Morris, A., and Thayer, S. (2006b). Topological exploration of subterranean environments. *Journal of Field Robotics*, 23:395–415.

[Stachniss et al., 2005] Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using rao-blackwellized particle filters. *Robotics: Science and Systems*.

[Tabib et al., 2016] Tabib, W., Corah, M., Michael, N., and Whittaker, R. (2016). Computationally efficient information-theoretic exploration of pits and caves. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3722–3727.

[Teledyne optech, 2019] Teledyne optech (2019). Cavity monitoring system. `https://www.teledyneoptech.com/en/products/static-3d-survey/cms/`. [Online; accessed 5-October-2019].

[TheWorldCounts, 2020] TheWorldCounts (2020). Health effects of mining. `https://www.theworldcounts.com/challenges/planet-earth/mining/health-effects-of-mining`. [Online; accessed 21-July-2020].

[Thrun, 1998] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71.

[Tian et al., 2019] Tian, J., Wang, Y., and Yuan, D. (2019). An unmanned aerial vehicle path planning method based on the elastic rope algorithm. In *IEEE International Conference on Mechanical and Aerospace Engineering*, pages 137–141.

[Tripicchio et al., 2015] Tripicchio, P., Satler, M., Dabisias, G., Ruffaldi, E., and Avizzano, C. A. (2015). Towards smart farming and sustainable agriculture with drones. In *International Conference on Intelligent Environments*, pages 140–143.

[Vaishnav and Patil, 2018] Vaishnav, P. and Patil, R. J. (2018). A comparative study of ga and sa for solving travelling salesman problem. *International journal of engineering research and technology*, 3.

[Vanegas et al., 2019] Vanegas, F., Gaston, K. J., Roberts, J., and Gonzalez, F. (2019). A framework for uav navigation and exploration in gps-denied environments. In *IEEE Aerospace Conference*, pages 1–6.

[Véras et al., 2019] Véras, L. G. D. O., Medeiros, F. L. L., and Guimaráes, L. N. F. (2019). Systematic literature review of sampling process in rapidly-exploring random trees. *IEEE Access*, 7:50933–50953.

[Wettach and Berns, 2010] Wettach, J. and Berns, K. (2010). Dynamic frontier based exploration with a mobile indoor robot. In *International Symposium on Robotics and German Conference on Robotics*, pages 1–8.

[Woeginger, 2001] Woeginger, G. J. (2001). Exact algorithms for np-hard problems: A survey. *Combinatorial Optimization*.

[Worboys, 1986] Worboys, M. (1986). The travelling salesman problem (a guided tour of combinatorial optimisation). *The Mathematical Gazette*, 70(454):327–328.

[Xiao et al., 2018] Xiao, X., Dufek, J. S., Suhail, M., and Murphy, R. R. (2018). Motion planning for a uav with a straight or kinked tether. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8486–8492.

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151.

[Yang et al., 2014] Yang, L., Qi, J., Xiao, J., and Yong, X. (2014). A literature review of uav 3d path planning. In *World Congress on Intelligent Control and Automation*, pages 2376–2381.

[Yuvka et al., 2020] Yuvka, S., Duran, E., and Uysal, O. (2020). Improvement an cost analysis of blasting operations at western lignite open cast mine. *Journal of scientific reports*, 0(44):44–56. Number: 44.

[Zammit and Van Kampen, 2021] Zammit, C. and Van Kampen, E.-J. (2021). Comparison between a* and rrt algorithms for 3d uav path planning. *Unmanned Systems*, 10.