UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Héctor Ignácio Azpúrua Perez-Imaz

Terrain-Aware Autonomous Exploration of Unstructured Confined Spaces

Belo Horizonte
2022

Héctor Ignácio Azpúrua Perez-Imaz

Terrain-Aware Autonomous Exploration of Unstructured Confined Spaces

Final version

Dissertation presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Douglas G. Macharet
Co-Advisor: Mario F. M. Campos

Belo Horizonte

2022

Héctor Ignácio Azpúrua Perez-Imaz

Exploração autônoma de espaços confinados não estruturados com terrenos irregulares

Versão Final

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Douglas G. Macharet
Coorientador: Mario F. M. Campos

Belo Horizonte

2022

Azpúrua Pérez-Imaz, Héctor Ignácio.

A996t Terrain-aware autonomous exploration of unstructured confined spaces [manuscrito] / Héctor Ignácio Azpúrua Pérez-Imaz. — 2022.
152 f. il.; 29 cm.

Orientador: Douglas Guimarães Marechat.
Coorientador: Mário Fernando Montenegro Campos.
Tese (doutotrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação
Referências: f.130-144.

1. Computação – Teses. 2. Robótica móvel -Teses. 3. Planejamento em ambientes 3D – Teses. 4. Exploração em espaços confinados – Teses. I. Marechat, Douglas Guimarães. II. Campos, Mário Fernando Montenegro. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. IV.Título.

CDU 519.6* 82.9(043)

Ficha Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6/1510 Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

TERRAIN-AWARE AUTONOMOUS EXPLORATION OF UNSTRUCTURED CONFINED SPACES

HÉCTOR IGNÁCIO AZPÚRUA PÉREZ-IMAZ

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Douglas Guimarães Marechat - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Mário Fernando Montenegro Campos - Coorientador
Departamento de Ciência da Computação - UFMG

Prof. Luiz Chaimowicz
Departamento de Ciência da Computação - UFMG

Prof. Gustavo Medeiros Freitas
Departamento de Engenharia Elétrica - UFMG

Prof. Gustavo Pessin
Instrumentação, Controle e Automação de Processos de Mineração - Instituto Tecnológico Vale

Prof. Denis Fernando Wolf
Instituto de Ciências Matemáticas e de Computação - USP

Belo Horizonte, 08 de julho de 2022

Documento assinado eletronicamente por **Douglas Guimaraes Macharet**, **Professor do Magistério Superior**, em 03/08/2022, às 15:46, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Mario Fernando Montenegro Campos**, **Professor do Magistério Superior**, em 03/08/2022, às 15:50, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Gustavo Medeiros Freitas**, **Professor do Magistério Superior**, em 03/08/2022, às 21:35, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Chaimowicz**, **Professor do Magistério Superior**, em 04/08/2022, às 11:22, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Gustavo Pessin**, **Usuário Externo**, em 04/08/2022, às 15:48, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Denis Fernando Wolf**, **Usuário Externo**, em 05/08/2022, às 15:17, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1650781** e o código CRC **2D2456E7**.

Referência: Processo nº 23072.246149/2022-17 SEI nº 1650781

Dedicado a mi familia, a mi padre y a mi abuela con mucho cariño.

Acknowledgments

So many people and institutions helped me in some way through developing this work, so probably someone will be missing; please forgive me! Ten of the twelve years I have been living in Brazil have been, in some way, related to the VeRLab laboratory. I'm very thankful to Brazil for the opportunity to make a life here, get a master's degree with a scholarship, and now get a Ph.D. in one of the best universities in Latin America. I'm also thankful to my advisors, Douglas and Mario, who gave me the chance to learn and live the world of research –something I didn't know before. To Chaimo and Anderson, thank you for the support; my first paper (best paper award!) was with you. To the friends I made along the way at the UFMG, thank you, I could not make it this far without the support, laughs, cachaça, and exciting chats with you guys. I still miss our card games at the cafeteria. To the guys at CORO, thanks as well. We competed against the best in the world at the XQuad team! That was super cool.

I greatly thank Vale S.A. and the Instituto Tecnologico Vale (ITV) for the fantastic opportunity to work making industrial robotics applications. Thanks to Pessin, Amilton, and Uzeda for believing in my work. A special thanks to Gustavo Freitas; I learn a lot from you professionally and as a friend. To the guys at NASA JPL, Maira, Ali, and Thiago thank you for allowing me to participate in the DARPA SubT challenge; it was amazing, even remotely. To the people I advised, Magno, Jacô, Mauricio, Andre, and Cota, thank you for believing in me and placing your trust in me. I'm sure that I learned more from you than vice-versa. Also, I thank the Komatsu Group and Modular Mining Systems for their support these last couple of months.

I thank my family for encouraging me from afar and my dogs for their unconditional love. A massive thanks to my best friend (and wife), Daniela. She believed in me when no one did, stood by me through all those years (+12) of wins and losses, and even in the darkest hour, is there for me. I love you so much! This is for us!

To all the crazy scientists who inspired me to get my degree: Dr. Doom, Dr. Manhattan, Tony Stark, Dexter, and Dr. Frankenstein, thank you!

Thank you all!

"Siempre respeté sus ideas, hasta que empezó a llevarme la contraria"

(Zapata)

Resumo

Esta tese aborda o problema de explorar ambientes confinados de forma autônoma usando robôs móveis terrestres. Aqui propomos um método de planejamento de caminhos em terrenos tridimensionais acidentados e duas técnicas de exploração que utilizam as áreas navegáveis do mapa, o custo de navegação relacionado e as informações esperadas de uma fronteira para selecionar o próximo setor de exploração. Os pipelines apresentados são construídos sobre uma representação do ambiente baseada em meshes tridimensionais e nuvens de pontos. Os meshes reconstruídos são convertidos em grafos de traversabilidade, e as regiões perigosas ou não transitáveis são filtradas considerando a pose esperada do robô no momento do planejamento. A geração dos caminhos mais promissores utiliza uma combinação linear de pesos aplicados a múltiplas métricas de atravessabilidade do terreno, como distância, rugosidade e consumo de energia do robô. O método de exploração proposto utiliza as informações volumétricas esperadas das fronteiras após simular a sua visitação; desta forma, as áreas de exploração são selecionadas de acordo com sua utilidade esperada (ganho de informação) e custo de visita, considerando a topografia do terreno. Também propomos um método de planejamento online usando diretamente a nuvem de pontos capaz de desviar de obstáculos em ambientes dinâmicos. A fase online é mais rápida para calcular e usa um algoritmo RRT com viés para as fronteiras mais informativas, sendo capaz de planejar e selecionar uma fronteira de destino sem especificar uma métrica de utilidade determinística. Os algoritmos foram validados em ambientes representativos simulados e reais, comprovando a viabilidade das metodologias propostas.

Palavras-chave: Robótica Movel, Planejamento em ambientes 3D, Exploração em espaços confinados, Robótica industrial.

Abstract

This dissertation addresses the problem of exploring confined environments autonomously using terrestrial mobile robots. Here we propose a methodology for path planning in rough three-dimensional terrains and two exploration techniques that use the map's navigable areas, the related navigation cost, and the information expected from a frontier to select the next exploration sector. The presented pipelines are built over a representation of the environment based on three-dimensional meshes and raw point clouds. The reconstructed meshes are converted into traversable graphs, and the dangerous or untraversable regions are filtered considering the expected pose of the robot at the time of planning. The generation of the most promising paths uses a linear combination of weights applied to multiple traversability metrics of the terrain, such as distance, roughness, and energy consumption of the robot. The proposed exploration method uses the expected volumetric information of frontiers after visitation; this way, the exploration areas are selected according to their expected utility and visit cost, considering the terrain's topography. We also propose an online planning phase using the raw point cloud to avoid obstacles in dynamic environments. The online phase is faster to compute and uses an RRT algorithm biased towards the most informative frontier, which is capable of planning and selecting a target frontier without specifying a hardcoded utility metric. The algorithms were validated in representative simulated and real environments, proving the viability of the proposed methodologies.

Keywords: Mobile Robotics, Exploration, Path planning in 3D environments, Confined Spaces, Industrial Robotics.

List of Figures

List of Tables

List of Algorithms

Acronyms List

AI Artificial Intelligence

APF Artificial Potential Field

BVP Boundary-Value Problem

CAD Compter Assisted Design

DoF Degrees of Freedom

GP Gaussian Processes

IMU Inertial Measurement Unit

ITV Instituto Tecnológico Vale

LeGO-LOAM Lightweight and Ground-Optimized Lidar Odometry And Mapping

LiDAR Light Detection and Ranging

MPC Model Predictive Control

PoV Point of View

RADAR Radio Detection and Ranging

RRT Rapidly-exploring Random Tree

RRT* Optimum Rapidly-exploring Random Tree

ROS Robot Operating System

RTAB-Map Real-Time Appearance-Based Mapping

SfM Structure from Motion

SLAM Simultaneous Localization and Mapping

SLSQP Sequential Least SQuares Programming

SoNAR Sound navigation and ranging

TDoA Time Difference of Arrival

ToF	Time-of-Flight
TWR	Two Way Ranging
UWB	Ultra-Wideband

Symbol List

$\lvert\, . \,\rvert$	Cardinality of a set
$\lvert\lvert\, . \,\rvert\rvert$	Euclidean norm
(a, b)	Linear regression parameters
α	Robot's roll pose
β	Robot's pitch pose
$\mathcal{B}$	Mesh $\hat{\mathbb{M}}$ frontiers
$C_1(p)$	Cost function for euclidean distance over the path p
$C_2(p)$	Traversability cost function over the path p
$C_3(p)$	Cost function for energy consumption over the path p
$\mathcal{C}$	Cluster of reachables frontiers in $\mathcal{G}$
$\mathcal{C}_{min}$	Minimum cluster group size
$\mathcal{E}$	The environment
E	Edges of a graph
eps	Minimum distance to create a cluster
$\mathbb{F}$	Collection of mesh $\mathbb{M}$ faces
$\mathcal{F}$	Collection of reachable frontiers
γ	Robot's yaw pose
$\mathcal{G}$	Traversability graph
$h(x, y)$	Terrain interpolation function
$\mathcal{H}$	Convex hull of the spherical projected points in the visibility filtering algorithm
$\mathbb{M}$	Reconstructed mesh of the environment
$\hat{\mathbb{M}}$	Mesh without the faces with a greater $\vec{\mathbf{n}}^z$ angle that θ_{max}
μ	Mean of a Gaussian distribution

$\mathcal{M}$	Three-dimensional occupancy grid representation of $\mathcal{E}$
$\hat{\mathcal{M}}$	Filtered version of $\mathcal{M}$ with the traversable points only
n_{goal}	Goal node in the traversability graph $\mathcal{G}$
n_{start}	Start node in the traversability graph $\mathcal{G}$
$\vec{\mathbf{n}}^z$	Normal vector of node at the z axis (face or point z normal)
$N_{\mathbb{F}}(i)$	The set of neighboring faces of face i in $\hat{\mathbb{M}}$
$N_{\{d,t,e\}}$	Normalization coefficients for the combined metric different components
o	Closest non-traversable point
$\mathcal{O}$	Big O asymptotic notation
p	A traversable path for ground robots
p_{global}	Estimated global traversable path for ground robots
p_{local}	Estimated local traversable path for ground robots
$x_{sampled}$	Point inside this radius closer to the target frontier
$\bar{p}$	Modified path p
$P_{\{d,t,e\}}$	Weights for the combined metric different components
$\mathcal{PC}$	Raw point cloud from the LiDAR sensor
q	Instant Pose of the robot R
$\mathbf{q}_{\mathrm{i}}$	Initial robot configuration
$\mathbf{q}_{\mathrm{g}}$	Goal configuration
r	Random value in the range $(0, 1]$
R	Representation of a robot (real or simulated)
$\hat{\mathcal{R}}$	Updated robot $\mathcal{R}$ pose
$\mathbb{R}^3$	Coordinate space over the real numbers (3 dimensions)
$\mathcal{SP}$	Support polygon of the robot
$sim(a, b)$	Similarity function based on the RMSE between two pointclouds a and b
σ^2	Variance of a Gaussian distribution
σ	Standard deviation of a Gaussian distribution
SE(3)	The Special Euclidean Group in 3 dimensions
$\mathcal{T}$	Octree of the map $\mathbb{M}$
τ_{bump}	Bumpiness threshold
$\tau_{inflation}$	Border inflation threshold in $\mathcal{G}$

τ_{iter}	Threshold on the maximum number of iterations
τ_{quat}	Threshold for quaternion angular displacement
$\tau_{terrain}$	Threshold to select the terrain interaction model given the neighbors angles
τ_{radius}	Threshold for the visibility radius
τ_{random}	Threshold to select a random point in the RRT algorithm
τ_{target}	Threshold to detect if the path target was reached
τ_z	Threshold for z linear displacement
θ_{max}	Maximum slope angle traversable by the robot
θ	Angle between to nodes or vectors
V	Vertices of a graph
$\mathcal{W}$	A set of weights used for weighted averaging
$\vec{Z}$	Canonical Z axis vector
$\mathbb{X}$	Untraversable set of points in $\hat{\mathcal{M}}$
x_{biased}	Selected point for exploring the RRT tree
$x_{nearest}$	Nearest unexplored point of x_{biased} in $\hat{\mathcal{M}}$
x_{new}	The resulting point of the *steer* section of the RRT algorithm

Contents

Chapter 1

Introduction

"You know better than to trust a strange computer!"

— C-3PO.

The undeniable benefit robots bring in security, reproducible performance, speed, and overall efficiency are some of the paradigms of the fourth industrial revolution or Industry 4.0. In this current industrial revolution, where most mechanical devices are digital and interconnected, robots can integrate seamlessly with multiple cyber-physical systems, allowing autonomous and independent decision-making using their embedded algorithms and Artificial Intelligence (AI) [Lasi et al., 2014]. The ongoing automation of traditional industry or manufacturing applies mainly to large static industrial robots. However, as we deepen into industry 4.0, new initiatives using smaller mobile robots are progressively being applied in a vast range of applications, such as mapping, monitoring, surveillance, and exploration. Three-dimensional robotic exploration has arisen as a challenging problem given the complex environments and situations where real robot deployments are needed, such as confined and GPS-denied environments.

1.1 Contextualization

The robotic exploration problem is related to the coverage problem, where the aim is to cover a known region with the more efficient method available (optimizing time, speed, or battery, for example). When performing exploration, the map is not known a priori, meaning that the robot will need to build a map and define where to go among a frontier of the known and unknown [Yamauchi, 1997]. Effective exploration strategies will generate a complete map, or close to it, in a reasonable amount of time. In general, traditional exploration methods focus on outdoor areas or single floor indoor scenarios that are far from the reality of prevalent industrial environments, such as confined subterranean caves and tunnels. These confined spaces present very different and challenging operational conditions for robots than the well behaved and structured indoor spaces for regular human use. Irregular terrains, steep slopes, tight and closed spaces, poor wireless communication, magnetic interference, lack of GPS signal, and slippery grounds are frequent features in many of those environments [Morris et al., 2006], thus rendering traditional exploration methods unfeasible.

Exploration is also closely related to mapping, in the sense that to efficiently explore an unknown region, the robot will need to keep an approximate localization estimation and generate a map to keep track of the already visited areas. So, to perform a proper exploration of an unknown environment, having a reliable map and localization estimation is critical.

1.2 Motivation

Numerous modern buildings have a significant volume of underground industrial infrastructure such as sewers, pipes, caves, and other urban underground structures including subway tunnels [Martz et al., 2020]. Underground structures are more significant in industrial settings, such as the mining industry, dealing with both human-made and natural-made tunnels and caverns. Although the definition of a confined space can vary depending on legislation and country, it is generally recognized that many underground environments can be classified as confined spaces, i.e., areas where the entrances and exits are limited, not designed for continuous human occupancy, and where the ventilation systems are insufficient to oxygenate the interior or remove any contaminants from the air [Ministry of Labour, 2006; Regulations, 1997; Government of Ontario and Branch, 2020]. Subterranean and confined spaces can also be categorized as structured or unstructured. Most industrial and civilian buildings are structured, predictable environments with few uncontrolled variables, allowing a robot to know what to expect when navigating through them. However, in this work, we also consider unstructured environments such as natural caves, caverns, or disaster scenarios, which are more challenging to predict, requiring a robot to identify and adapt to environmental changes and variables. Figure 1.1 shows a high-level classification of the characteristics of confined and subterranean environments.

Some risks related to confined spaces are the presence of venomous animals, noxious gases or excrement, extreme temperatures, narrow spaces, unhealthy oxygen levels, flooding, and collapsing structures, among others. Representative examples of

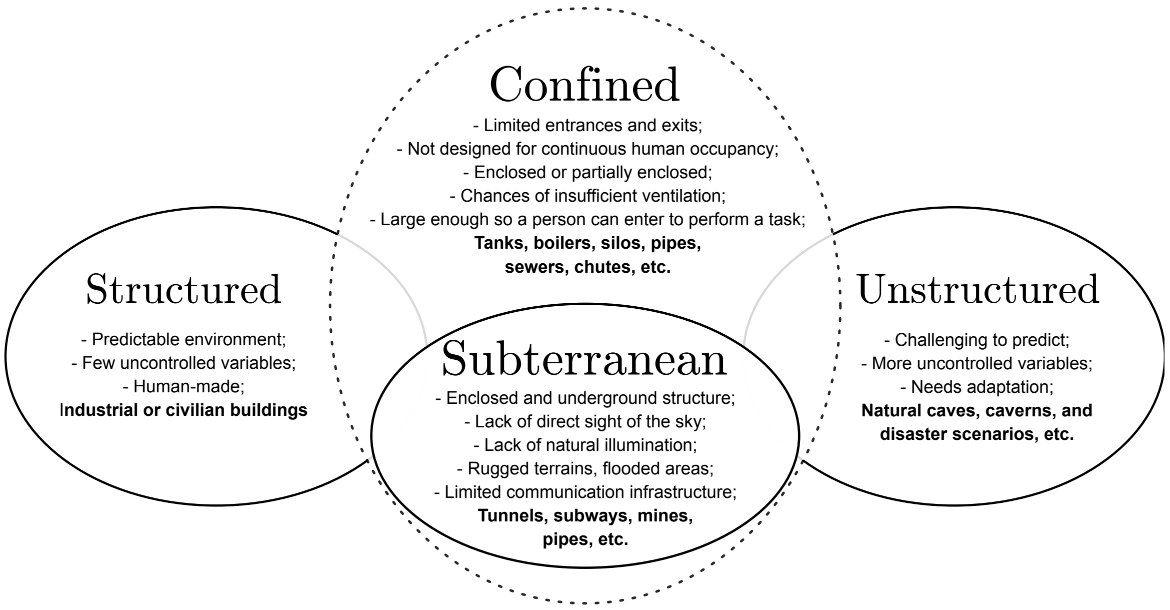

Confined
- Limited entrances and exits;
- Not designed for continuous human occupancy;
- Enclosed or partially enclosed;
- Chances of insufficient ventilation;
- Large enough so a person can enter to perform a task;
Tanks, boilers, silos, pipes, sewers, chutes, etc.
Structured
- Predictable environment;
- Few uncontrolled variables;
- Human-made;
Industrial or civilian buildings
Unstructured
- Challenging to predict;
- More uncontrolled variables;
- Needs adaptation;
Natural caves, caverns, and disaster scenarios, etc.
Subterranean
- Enclosed and underground structure;
- Lack of direct sight of the sky;
- Lack of natural illumination;
- Rugged terrains, flooded areas;
- Limited communication infrastructure;
Tunnels, subways, mines, pipes, etc.

Figure 1.1: High-level classification of the characteristics of confined and subterranean spaces.

confined underground spaces are shown in Figure 1.2.

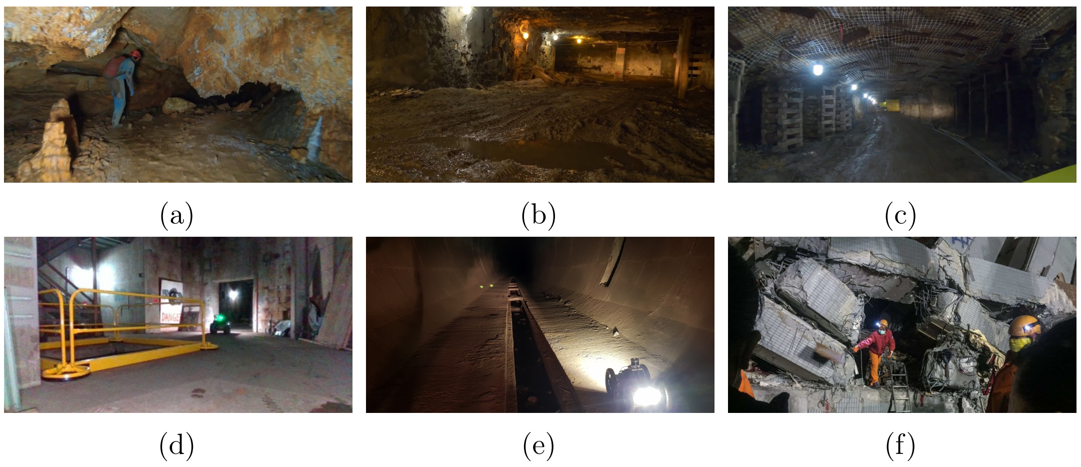


Figure 1.2: Examples of confined subterranean spaces: caves (a), industrial tunnel systems (b-c), and an urban scenario (d) from the DARPA Subterranean Challenge [DARPA, 2019, 2020], a subway-like structured tunnel system (e) and a collapsed building (f) [Bradsher, 2016].

A particular challenge for ground robots in subterranean or enclosed scenarios lies in the terrain topography, which is commonly complex and unstructured, presenting a mix of flat and rugged areas (Figure 1.3). These particular characteristics require any exploring agent to have efficient locomotion and navigation systems capable of overcoming obstacles while also considering energy consumption and payload capabilities.

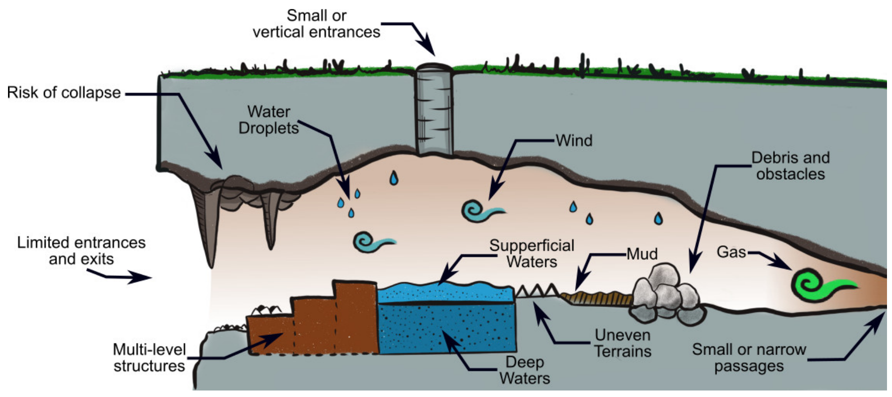


Figure 1.3: Extension of the illustrative scenario presented in [Martz et al., 2020]. Known terrain challenges for robotic locomotion within a subterranean mine include limited entrances and exits, mud, presence of water bodies, uneven, rocky ground, rock piles and debris, strong wind gusts, gas, and dripping water from the ceiling, among others.

Some of the most demanding tasks in multiple industrial scenarios are inspection and mapping. Inspections are needed to assess the health of structures and equipment such as pipes, dams, mills, and sewers, among many. In other cases, inspections are needed for non-structured outdoor environments, such as natural caves and open-pit mines. Caves are a particular challenge in the mining industry since environmental legislation requires continuous assessing and monitoring of those natural environments. Any environmental problems detected in an operational site have the potential of hindering or entirely stopping the mineral exploration of a mining complex.

Despite photos and videos are useful for inspectors to generate an initial assessment of a particular situation, 2D information has its limitations. Inspectors could benefit from using accurate 3D representations, allowing the analysis of geometrical information and a spatial relationship with an increased immersive experience. The detection of obstacles, debris, blockades, and the assessment of the structure changes through time, is also facilitated using 3D data.

Currently, many of those inspections are performed manually by human operators carrying sensors, making the task prone to measurement errors, lack of consistent performance, and made the operators vulnerable to environmental-related dangers. Natural and industrial confined spaces are known to have the possibility of toxic or flammable gases, lack of oxygen, poisonous animals, roof collapse, risk of falling from high-level platforms, among others.

The mining industry, one of the highest-grossing industries in the world—and the main scenario where this thesis focuses on— have millions of workers worldwide, and unfortunately, this industry holds a high number of accidents, some of them fatal [Hull et al., 1996; Mitchell et al., 1998; Sanmiquel et al., 2018]. Only at the Brazil operations of Vale S.A, one of the biggest international mining companies (and the largest iron ore extractor worldwide), there are more than 73.000 active employees. Therefore, reducing the chances of human endangering is a constant concern. In this sense, Vale S.A and Instituto Tecnológico Vale (ITV) developed the EspeleoRobô platform, a terrestrial robot to inspect industrial confined areas, which is used as the main robotic platform in this work (Appendix A describes the EspeleoRobô platform in detail).

It is a challenge to measure the economic impact of worker's injuries since the costs of specific occupational injuries in mining are not openly available, as mining and insurance companies do not usually share this information. Nevertheless, as stated by Heberger [2018], there are non-negligible economic gains by augmenting worker's safety, thus making safety a priority also from an economic point of view.

Likewise, in an emergency scenario, another type of workers, such as First Responders, are also at risk when performing search and rescue during a disaster (Figure 1.4). First responders are specially trained personnel who are among the first to arrive and assist at the emergency scene, such as an accident, natural disaster, or terrorism. In this case, it is especially important to thoroughly explore the environment since not always an accurate map is available, and the location of injured people could not be pinpointed to particular areas in the map. Therefore, first responders could also benefit from the use of autonomous 3D map generation with robots. Many disaster scenarios could also classify as a confined environment [Murphy, 2014]. The lack of consistent communication channels, complex topography, unavailability of any global localization, and severe health risks make these environments very challenging to explore too.

(a)

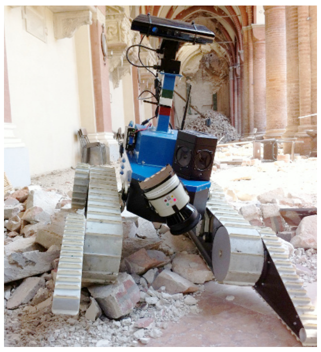

(b)

Figure 1.4: First responders could receive help from multiple coordinated robots for exploration and inspection in dangerous situations such as: (a) victim search in complex indoor spaces*, (b) and inspection after an earthquakes or other disasters†.

Given those priors, a reliable way to decrease risks for human industrial operators and first-responders alike is to remove them from the dangerous areas by using mobile robotic devices instead. However, despite the advances in the robotics field, there are still challenges to overcome, especially in robustness, task fulfillment, and resiliency.

1.3 Problem

One of the most significant challenges of robotic exploration in confined spaces is related to the robot locomotion capabilities and its interaction with the ground topography, including obstacles. Until today there are multiple proposals for flexible and usable navigation pipelines for real rugged scenarios. However, most of these proposals are dependant on platform type and are tailored for the specific scenario, and generally consider only open spaces.

An option to circumvent many traditional robots' locomotion pitfalls and limitations could be using only aerial platforms instead of ground robots. However, given the limited payload, flight time, and overall fragility of aerial platforms today, terrestrial platforms still can be seen as desirable platforms for use in extended areas or rugged environments. Hence, the first step needed to perform autonomous robotic exploration in complex environments is a robust navigation method that considers the topography in multi-level and complex scenarios. Many of the industrial environments studied in this dissertation have holes, slopes, and many other types of obstacles that render 2D path planning and navigation impracticable. For ground robots in real industrial scenarios, online path estimation over three-dimensional maps is a practical requirement. Therefore, in this work, we propose efficient path planning techniques that could be executed online, directly on robot hardware, and consider metrics such as terrain roughness, energy consumption, and distance.

The complete exploration process is highly dependant on a good map and localization estimations. Traditional two-dimensional SLAM algorithms work well for structured, indoor scenarios. However, other types of mapping and localization are needed for confined, unstructured, and rugged environments, especially considering the lack of global positioning. Highly representative maps are needed to plan detailed movements to overcome obstacles and other pitfalls. Therefore, as the final exploration

*`https://www.newsweek.com/2017/03/03/nasa-pointer-tracking-system-first-responders-558689.html`

†`https://horizon-magazine.eu/article/robot-rescuers-help-save-lives-after-disasters.html`

objective is to generate some map representation, we consider using 3D maps and SLAM algorithms as input for the exploration pipeline.

The traditional way of exploring an environment is to iteratively visit frontiers –the boundaries of the explored and unexplored. Detecting and estimating the best next frontier is an open challenge since it depends on many factors. A greedy approach to selecting the next biggest frontier is not always a guarantee of global optimality. Short-sighted methods could not detect that geometrically larger frontiers could bring little to no informational gain after performing a long journey to visit them, expending precious energy navigating into low-rewards goals. Therefore, energy consumption, traversable terrain difficulty, path length, and informational gain of a frontier could be used simultaneously to estimate the best global next frontier to visit.

A high-level definition of the problem tackled in this dissertation could be defined then as:

Autonomous exploration of confined spaces with rugged terrain by a mobile robot: *given a terrestrial robot in a 3D confined environment, the objective is to efficiently perform a complete exploration in a way that optimizes multiple metrics such as terrain traversability, time, energy consumption or vehicle safety.*

A description of a sequential process of exploration over a complex 3D cave environment, as proposed in this dissertation, can be observed in Figure 1.5, where a robot begins in an unknown position of an unknown map and, given the sensory input of a 3D sensor, it estimates the best action to maximize the exploratory gain towards the environment.

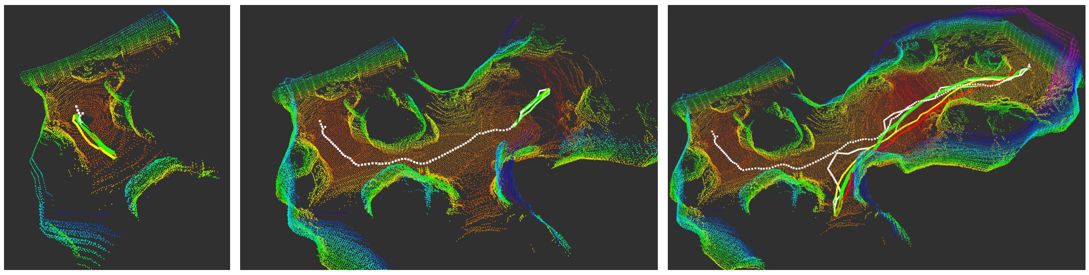


Figure 1.5: Sequential process of exploration in a complex 3D cave environment with a terrestrial robot (left to right). The dotted white line is the robot odometry, and the colored solid lines are possible paths for the robot to take given a metric of efficiency.

From this perspective, this work presents a group of methods and techniques for dealing with mobile robotic exploration in confined spaces. Our study is divided into multiple sequential parts: (i) an efficient method for 3D navigation in rugged terrains, (ii) a method for autonomous and deterministic 3D terrain-aware exploration, and (iii) a methodology for efficient probabilistic exploration with local obstacle avoidance.

Our exploration and path planning methods use meshes and point clouds to represent terrain roughness and traversability. We use a reconstruction methodology tailored for confined and subterranean spaces to generate the mesh. In contrast to the state-of-the-art in subterranean exploration, in this work, we propose using a limited set of sensors: a single 3D LiDAR, IMU, and wheel odometry. This sensor configuration allows us to overcome the illumination challenges of confined spaces, including partial fog and smoke, while being light and energy efficient. Regarding exploration, we present two pipelines: deterministic and probabilistic. The deterministic method uses optimal path planning techniques and metrics to estimate the next frontier to visit, however, its limited to static environments. The probabilistic one uses a modified version of Rapidly-exploring Random Tree (RRT), which we call MI-RRT, to plan and select the frontier simultaneously, capable of dealing with dynamic obstacles.

1.4 Objectives

Our main objective in this work is to propose a methodology for a single terrestrial robot to perform exploration missions in confined industrial scenarios. We understand robotic exploration as the action of covering an unknown region in the most optimized way possible, given a performance metric. To achieve this objective, we need to accomplish the following specific objectives:

1. Perform safe and cost-effective navigation in confined and rugged 3D scenarios;
2. Use efficient methods for 3D robotic exploration and frontier selection;
3. Allow obstacle avoidance of unexpected and dynamic environmental changes.

1.5 Contributions

The main contributions of this dissertation can be summarized as follows:

1. Chapter 2 shows an in-depth revision of the recent works and results related to the autonomous exploration of confined spaces. This contribution focused on Objectives 1, 2, and 3: we need to know and understand the state-of-the-art to highlight which specific ideas are worth pursuing. For this, we prepared a survey, which is partially embedded in Chapter 2, describing critical areas of exploration in confined spaces, such as the characteristics and challenges of the confined environment, appropriated sensorial kit, path planning techniques in

rough terrain, traditional two-dimensional exploration methods, and three dimensional exploration. The strategies described in this chapter are the foundation of our proposed methodology.

2. Chapter 3 proposes a practical method for terrain-aware 3D navigation for terrestrial robots in confined and rugged spaces [Azpúrua et al., 2021b]. This contribution focused on Objective 1: an effective method for path planning and locomotion as a pre-requisite for exploration in complex 3D spaces with mobile robots. In this contribution, we focused on evaluating a complete pipeline for robotic navigation performing semi-autonomous exploration for subterranean spaces. In our proposal, the 3D point cloud generated by an online SLAM method is modeled as a graph with terrain-aware edge weights. The optimal path uses a combination of metrics, optimized by a graph search algorithm. In this sense, the pipeline allows visiting the waypoints selected manually by an operator and serves as the foundation for other autonomous approaches.

   Héctor Azpúrua, Adriano Rezende, Gilherme Potje, Gilmar Pereira, Rafael Fernandes, Victor Miranda, Levi Welington, Jacó Domingues, Filipe Rocha, Frederico Martins, Luiz Dias de Barros, Erickson Nascimento, Douglas G. Macharet, Gustavo Pessin and Gustavo M. Freitas **Towards semi-autonomous robotic inspection and mapping in confined spaces with the EspeleoRobô**. Journal of Intelligent & Robotic Systems, 2021. [Qualis-CC A2]

3. In Chapter 4, we present a methodology for autonomous 3D exploration in confined subterranean environments. This contribution is related to Objective 2: efficient 3D exploration and frontier selection Azpúrua et al. [2021a]. In this contribution, we proposed a new method for autonomous exploration that leverages the previous navigation scheme to generate more efficient traversable paths that consider static obstacles and narrow passages. We used the 3D sensor parameters to project the expected best view of a frontier and estimate all reachable frontiers' Information Gain. The next best frontier selected for visitation uses a tradeoff between the navigation cost and the estimated frontier information gain.

   Héctor Azpúrua, Mario F. M. Campos, and Douglas G. Macharet **Three-dimensional Terrain Aware Autonomous Exploration for Subter-**

ranean and Confined Spaces. International Conference on Robotics and Automation (ICRA 2021), 2021. [Qualis-CC A1]

Héctor Azpúrua, Mario F. M. Campos, and Douglas G. Macharet **Research Statement: Towards Terrain-Aware Autonomous Exploration in 3D Confined Spaces**. Robotics: Science and Systems 2021 (RSS 2021), 2021. **(Workshop presentation)**

4. Finally, in Chapter 5, we present a methodology for autonomous 3D exploration using an efficient probabilistic framework capable of online obstacle avoidance. This contribution is related to Objective 3: obstacle avoidance from unexpected and dynamic environmental changes. In this contribution, we proposed a new method for autonomous exploration using a biased RRT algorithm for global planning and a traditional RRT for local planning. Both sampling-based algorithms work by directly using the raw point cloud; hence they are faster than the previous exact approaches and can be executed online. Furthermore, the biased RRT, called MI-RRT, uses the expected information gain of the frontiers as biases for expanding the RRT tree, thus reducing the time spent in an exploration cycle by performing the frontier selection and path planning at the same step.

1.6 Organization

The following chapters of this dissertation are organized as follows. In Chapter 2, we present the related works in robotic exploration with ground robots, including the definition of confined spaces and the sensors most adequate for exploration in those environments. Chapter 3 shows our proposal for 3D navigation in complex environments and Chapter 4 presents our methodology for deterministic 3D robotic exploration in confined spaces. Chapter 5 shows a probabilistic exploration method that allows for local obstacle avoidance. Finally, our conclusions and future research directions are defined in Chapter 6.

Chapter 2

Related Work

"Nothing is static, everything is evolving, everything is falling apart."
— Tyler Durden.

Yamauchi—a pioneer in the field—defines autonomous robotic exploration as the act of "moving through an unknown environment while building a map that can be used for subsequent navigation" [Yamauchi, 1997]. Despite autonomous exploration with mobile robots is a popular subject, and a considerable amount of work is already available about it, exploring real-world industrial scenarios is still an open problem. Real-word robotic exploration requires robust and reliable map generation, precise localization, safe navigation, and efficient path planning. Those requirements make exploration a tough problem for complex 3D environments with rugged terrains. The challenge is then increased in confined environments, considering that many locations do not have proper networking or lighting infrastructure, previous map information is generally missing, global localization is unfeasible, and there are many locations with no proper and well-defined traversable areas.

2.1 Representation of the environment

Learning how to generate maps has been an important research area in robotics. Map generation with single robots requires the solution for three problems: mapping, localization and path planning. The mapping problem tries to answer "What does the world looks like?". In this sense, we could say that mapping is the problem of integrating the information gathered with the robot's sensors into a given representation. On the other hand, localization tries to answer "Where am I?", meaning finding the correct transformation between the robot and a given reference frame of the world. Finally, path planning refers to "How to reach that goal?", meaning the generation of paths that the robot could follow to reach a place. As seen in Figure 2.1 there are overlapping areas between those challenges that generate a new type of problems by itself, such as SLAM, active localization, and exploration.

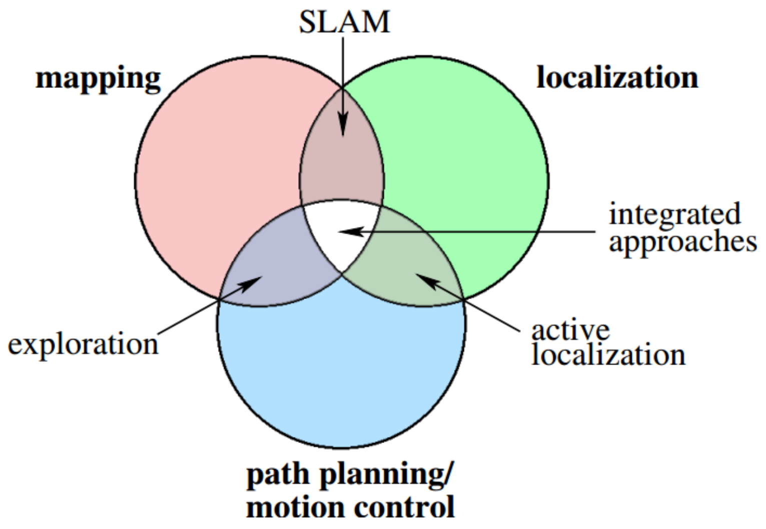


Figure 2.1: Challenges faced by a robot in need to acquire accurate models of the environment. In this regard, exists overlapping areas between those main challenges that generates a new type of problems by itself [Makarenko et al., 2002].

We know that to generate a proper map, we must have an accurate position estimation of the robot, but proper localization also needs a good map. This "chicken-and-the-egg" dilemma between mapping and localization derived into the Simultaneous Localization and Mapping problem (SLAM). SLAM algorithms aim to give a robot the capacity to build a global map of the visited environment and, at the same time, utilize this map to deduce its own location at any moment [Fuentes-Pacheco et al., 2015]. Active localization is related to the problem of moving the robot to places where it can improve its localization. Exploration then assumes an accurate localization to generate a map and gather information about an unknown environment. This work focus on the sub-problem of exploration, where the robots do not know the map *a priori*, and the path planning will need to guide the robot in an intelligent way to reduce the unknown areas of an environment.

2.1.1 Sensors for mapping and localization applications in confined spaces

The sensor suite for mapping and localization with an autonomous mobile robot inside confined and subterranean spaces requires careful evaluation given the possible harsh environment the equipment will face. Multiple works have studied and compared different sensor groups and technologies for subterranean spaces [Wong et al., 2011; Leingartner et al., 2016; Rauscher et al., 2016; Bijelic et al., 2018; Szrek et al., 2021]; however, to the best of our knowledge, a comprehensive usability analysis of the most widely used sensors for these particular scenarios does not exist. Sensor analysis and comparison in representative scenarios are fundamental, primarily since most manufacturers' sensor evaluation is performed in ideal situations and may not represent real-world robotic setups in harsh environments. In this sense, we do not intend to provide a comprehensive study of all possible sensor configurations or combinations, but rather a systematic analysis focused on a reasonable number of popular sensors appropriate for underground use.

Some of the most commonly used sensors for mapping in robotics are range sensors such as Light Detection and Ranging (LiDAR), radar, sonar, passive cameras (stereo or mono), and active cameras (RGB-D or Time-of-Flight (ToF)). Other types of sensors can be used for direct/indirect localization and improve the overall mapping procedure, such as Inertial Measurement Unit (IMU), encoders, or other relative localization systems such as Ultra-Wideband (UWB) beacons or related wireless technologies.

Planar LiDAR sensors (Hokuyo 30ULX) or multi-line LiDAR sensors (Velodyne VPL-16) are some of the most prevalent devices for accurately mapping an environment in robotic applications, given their robustness and accuracy [Wong et al., 2011]. Most LiDAR technologies work using the ToF principle, where a single or multipoint laser is rastered horizontally using a group of spinning mirrors generating a point cloud. For single-beam LiDAR, attaching a motor or actuator to tilt the sensor continuously allows the acquisition of 3D range data. Since ToF LiDAR sensors estimate distance by measuring the time a laser takes to travel from the sensor and return, they can only provide data on the angle, distance, and reflectivity of a surface. The main drawbacks of this technology are size, weight, low spatial resolution at a distance (sparse points), the lack of texture and the use of rotating parts in most current sensor models. LiDAR system performance can also be degraded by extreme environmental conditions such as dust and severe fog [Peynot et al., 2009; Rasshofer et al., 2011; Bijelic et al., 2018], which can be common in many confined underground spaces.

Sound navigation and ranging (SoNAR) is an active technique for robot sensing using ultrasound waves. By estimating the ToF of ultrasonic pulses, it is possible to estimate object distances with high accuracy [Akbarally and Kleeman, 1995]. This method allows for good short-range precision at the expense of spatial accuracy and is found to work efficiently in several restricted humid environments where LiDAR does not work [Silver et al., 2004; Fairfield et al., 2006]. Another active approach for range measurements is Radio Detection and Ranging (RADAR), which is an established technology for localization and obstacle avoidance in autonomous driving given its robustness to many environmental situations, and is also a feasible option for underground environments [Dickmann et al., 2014; Roos et al., 2019]. RADAR, like SoNAR, does not require light or temperature gradients to operate. Nevertheless, RADAR can be adversely affected by noise, spatial binning (resolution), and data corruption such as multipath reflections.

Monocular cameras can represent the world as a 2D matrix of pixels. These cameras can be monochrome or color (RGB), and a variable frame rate depends on the use case. Alone, these cameras cannot detect depth. However, with a calibrated camera and techniques such as Structure from Motion (SfM), it is possible to estimate a 3D representation of an environment [Bianco et al., 2018; Saputra et al., 2018]. Image information can also be used for landmark recognition. Planar images can help estimate motion and are generally used with other sensors such as LiDAR, encoders, or IMUs. A particular type of monocular camera called an event camera can provide information at a very high rate (microsecond resolution), with pixels responding asynchronously to changes in brightness [Gallego and Scaramuzza, 2017]. Typical subterranean environments lack natural or proper illumination, generating problems such as the absence of visible textures, shadows, rapid brightness changes when using external illumination, and occlusions that could render most passive imaging sensors unreliable.

In contrast to traditional monocular cameras, depth cameras can produce 3D or depth data in addition to traditional images. Some examples of such systems are stereo cameras, RGB-D cameras, and ToF matrix cameras. Stereo cameras are a particular arrangement of multiple monocular cameras with known extrinsic and intrinsic parameters that allow the passive extraction of 3D information from a scene. Active RGB-D cameras can give depth and color simultaneously, even in low light conditions. In this sense, some of the most popular RGB-D cameras are the Microsoft Kinect and Intel RealSense product lines. Active RGB-D cameras project patterns of structured infrared light into a scene and use the sensed deformation of the patterns of objects in the scene to accurately estimate depth [Dal Mutto et al., 2012]. Despite being more robust to a lack of proper lighting, active cameras have degraded performance in brighter outdoor environments.

ToF matrix cameras are a particular case of ToF sensors that use a 2D array of ToF pixels, therefore providing a reconstruction of a 3D surface with very high accuracy, outdoor reliability, and speed. Structured light sensors are more appropriate for close ranges (less than 4 m) given that the depth accuracy significantly decreases with distance [Scherer et al., 2012].

Close-range sensors such as bumpers, buttons, digital whiskers [Solomon and Hartmann, 2006], or other tactile sensing systems are not widely deployed in real-world subterranean robotic mapping applications and are mostly used for collision detection given their limited actuation range.

Odometry techniques are one of the most traditional localization methods, estimating the position of a mobile robot in time, using sequential sensor data relative to a specific coordinate system. The most popular sensors for odometry estimation in terrestrial robots are wheel encoders. These sensors can detect the rotational movement of a wheel and accurately determine how much it has turned. Encoder sensors can be mechanical, optical, magnetic, or work with electromagnetic induction. The rotational data alone can be used to estimate odometry via dead reckoning, but it is subject to cumulative errors and does not detect terrain problems such as slippery grounds. Usually, other sensors are needed to estimate an absolute position [Karlsson and Gustafsson, 2017]. In this sense, encoders and similar sensors are commonly used in conjunction with others such as LiDAR or IMUs to improve the overall robustness of Simultaneous Localization and Mapping (SLAM) or visual-odometry estimators [Shan and Englot, 2018; Labbé and Michaud, 2019].

IMUs are electronic devices that can measure the orientation, velocity, and gravitational forces of the object to which they are attached using one or multiple accelerometers, gyroscopes, and magnetometers. These sensors are generally used in conjunction with others such as LiDAR, encoders, or cameras to improve the quality of a robot's localization estimate [Yi et al., 2007; Brossard and Bonnabel, 2019]. These sensors can provide internal algorithms for filtering, delivering high accuracy pose estimations in short time windows. Odometry estimation using only IMU sensors is prone to drifting; thus, other sensors are also needed to provide robust absolute localization estimation.

Other types of localization use radio signals for direct or indirect position estimation. UWB is a type of short-range low-battery radio communication, capable of accurate distance measurements using the ToF principle. Two methods are used for localization: Time Difference of Arrival (TDoA) and Two Way Ranging (TWR), which are capable of an indoor and outdoor localization resolution up to 30 cm or less depending on the environment and sensor quantity [Marquez et al., 2017; Li et al., 2019]. This technique is robust to environmental situations such as low light, fog, rain, and typical signal problems of other wireless ranging solutions such as reflection.

In Table 2.1 we summarize the expected usability of many popular sensors used for map generation and exploration in confined and subterranean scenarios. The analysis and results are meant to guide sensor selection, given the perils and conditions of an underground space. Not all conditions can be considered; consequently, we selected the critical conditions from several research studies.

For this work, we selected a limited suite of sensors that could effectively deal with the roughness and illumination challenges of subterranean spaces: 3D LiDAR, IMU, and wheel odometry. The LiDAR can overcome multiple environmental situations such as fog and smoke better than image-based sensors. Furthermore, the IMU and wheel odometry can be integrated to improve the robot's localization while performing the SLAM, which experimentally has shown to be critical, especially where few geometrical features are available.

2.1.2 Map representations

The map representation used for a particular environment must be compatible with the sensor used and the mapping task's final goal: mapping accuracy or a less detailed map for navigation. In this sense, the most common map representations can be grouped into topological maps and metric/geometric maps. These approaches can also be divided into 2D and 3D map representations, where 2D maps are more suited for structured indoor spaces and 3D representations more suited for unstructured outdoor environments. In general, maps are used for navigation and environment reconstruction/analysis within confined spaces. The quality of a map will also reflect on the type of path planning performed: highly detailed maps allow for fine-grained navigation capable of avoiding obstacles, overhangs and reducing possible environmental hazards at the cost of extra memory and CPU consumption. In this sense, there exists a trade-off between map accuracy and the computational capability of a robot. Some map characteristics are best captured with fully 3D representations; however, there are cases where less complex maps will suffice, mainly when used as dynamic local obstacle maps.

A topological map is an abstract description of the structural characteristics of an environment. For example, topological maps can represent the connectivity of different places such as rooms, floors, or buildings connected by a sequence of robot actions. Generally, these types of maps are modeled as graphs, where nodes are locations, and arcs/edges are ways to reach them [Kuipers and Byun, 1988; Choset and Nagatani, 2001]. On the other hand, metric maps capture an environment's geometric properties,

Table 2.1: Hardware for mapping and localization in confined and subterranean environments

Hardware	**Example**	**Usability criteria in underground spaces**																		
		Non line-of-sight	High bandwidth	High precision	High resolution	Fast acquisition time	Remote sensing	Long/middle range sensing	Can capture 3D data	Rob. to poor illumination	Rob. to fog/etc.	Rob. to lack of texture	Rob. to drifting/mud	Rob. to water bodies	Lower CPU/GPU reqs.	Embebbed processing/filters	Rugged/Water resistant	Few/No moving parts	Lightweight/Small	Used in DARPA's SubT
Direct Localization[†]*																				
GPS	U-blox NEO-6M	○	-	○	○	○	○	○	-	○	○	○	○	○	●	●	●	●	●	○
UWB	DWM1001	◐	○	◐	◐	●	●	●	-	●	●	●	●	○	●	●	●	●	●	●
Wheel encoder	USDigital S1	-	-	◐	●	●	-	-	-	●	●	-	○	●	●	●	●	○	●	●
IMU/Accel./Gyro	Xsens MTi-610	-	-	●	●	●	-	-	●	●	●	●	○	●	●	●	●	●	●	●
Wireless Localization[†]*																				
Bluetooth	nRF52840	○	◐	○	◐	◐	●	◐	-	●	●	●	●	○	◐	○	●	●	●	●
ZigBee	Xbee S2C	○	◐	○	◐	◐	●	◐	-	●	●	●	●	○	◐	○	●	●	●	●
NFC/RFID	NXP PN532	○	◐	●	○	◐	◐	◐	-	●	●	●	●	○	◐	○	●	●	●	○
WiFi	Real. RTl8723DE	○	●	○	◐	◐	●	◐	-	●	●	●	●	○	◐	○	●	●	●	●
TTE (Through-The-Earth)	CanaryCommpac	●	◐	◐	◐	◐	●	●	-	●	●	●	●	○	◐	◐	●	●	○	○
Mapping/SLAM*																				
Event Camera	DAVIS346	-	-	●	◐	●	●	◐	○	◐	○	○	●	○	○	◐	●	●	●	○
Monocular RGB Camera	FLIR Firefly S	-	-	●	●	●	●	◐	○	○	○	○	●	○	○	○	●	●	●	●
IR ranging	GP2Y0A710K0F	-	-	◐	○	●	●	◐	○	●	○	●	●	○	●	●	◐	●	●	○
Passive 3D Camera (Stereo)	Stereolabs ZED	-	-	◐	●	●	●	●	●	○	○	○	●	○	○	◐	●	●	●	●
Touch	Bumpers/whiskers	-	-	●	○	●	○	○	○	●	●	●	●	●	-	-	●	○	●	●
Thermal/Spectral Camera	FLIR A700	-	-	◐	◐	◐	●	●	○	●	◐	●	●	◐	○	○	●	●	●	●
Sonar	MaxBotix MB1000	-	-	◐	◐	◐	●	○	○	●	●	●	●	●	●	●	◐	●	●	○
Radar	XM1321	-	-	◐	◐	◐	●	●	○	●	●	●	●	●	◐	◐	●	●	◐	●
Rotating sensor (LiDAR)	LiDAR + motor	-	-	●	●	●	●	●	●	●	◐	●	●	○	○	●	◐	○	○	●
Single-line LiDAR	Hokuyo 30LX	-	-	●	●	●	●	●	○	●	◐	●	●	○	◐	◐	●	◐	◐	○
Active 3D Camera (ToF, IR)	Intel D435i	-	-	●	◐	●	●	◐	●	●	○	◐	●	○	○	●	●	●	●	●
Multi-line LiDAR	Ouster OS1	-	-	●	●	●	●	●	●	●	◐	●	●	○	○	◐	◐	◐	◐	●

● = criterion fully covered; ◐ = partially covered criterion; ○ = non-covered criterion; "-" = not applicable;
[†]Only evaluated for its localization accuracy;
*Evaluated as standalone devices;

similar to a floorplan. The distinction between metric and topological maps can be considered fuzzy since most working topological approaches also rely on geometric information. In practice, metric maps are finer-grained than topological maps, but this comes with an extra computational cost [Thrun et al., 2002].

Occupancy grids were one of the first probabilistic map representations used by mobile robots [Moravec and Elfes, 1985]. These maps discretize an environment into small portions called grid cells, and every cell has information about the area it covers, such as the probability of cell occupation.

Occupancy grids have great representation capability because no prior information of the map or model of the environment is needed. Traditional occupancy grids are capable of multiple map resolutions but focus on only two dimensions. Other less popular 2D map representations are feature-maps [Newman et al., 2003] and point maps [Lu and Milios, 1997].

Thrun [1998] developed a hybrid metrical and topological approach. In this case, metric grid maps are learned using artificial neural networks and naive Bayesian integration and topological maps are generated by partitioning the grid map into regions. The author claims that by combining both paradigms the accuracy advantages of grid maps are improved by the efficiency of the simpler topological maps.

If an environment can be modeled as a height function $h = f(x, y)$ where x and y define a point in the environment and h is the height, then elevation maps or 2.5D maps are a reasonable choice. In an elevation map, the height value is stored in a discrete location, meaning that open and single-level scenarios can be represented efficiently given that no other vertical surface or overhangs exist. Elevation maps are common in planetary exploration given the certainty of no vertical structures other than the ground floor [Maimone et al., 2007]. Fankhauser and Hutter [2016] developed an efficient method to generate elevation maps or 2.5D maps, called GridMap (Figure 2.2). The method is fast enough for use in real-time surface reconstruction and terrain interpretation for uneven terrain. It supports a multiple layer system for different mapping metrics, such as altitude, angle, and roughness. It is currently used for multi-legged robot navigation, such as the ANYmal platform [Fankhauser and Hutter, 2018]. In [Yang et al., 2018], the authors present a hybrid 2.5D map representation called 2.5D-NDT, which simplifies a 3D occupancy grid into an elevation map of traversable areas.

Although 2D and 2.5D representations can work correctly in open areas, a full 3D model of the environment is needed when the robots can also move under or above the terrain. A point cloud can represent occupied points over a 3D environment with great detail, but it is a suboptimal representation that is not generally used directly for real-time planning. A significant number of robotics applications require a fast and

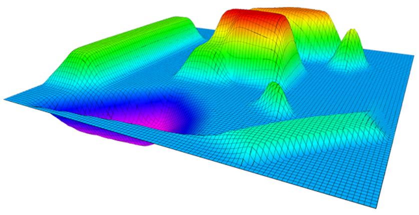


Figure 2.2: Grid map example of an elevation map [Fankhauser and Hutter, 2016].

memory-efficient probabilistic map representation, with the capacity of representing free and occupied areas. In this sense, one of the most popular techniques for probabilistic 3D mapping is OctoMap, which uses efficient modeling based on Octrees to minimize space and memory [Hornung et al., 2013]. Octrees are an efficient tree-type data structure in which each internal node has exactly eight children, commonly used for partitioning a three-dimensional space by recursively subdividing it into octants [Meagher, 1982]. In this regard, OctoMap can model arbitrary environments without prior assumptions about them, allowing empty, free, and unexplored spaces with arbitrary size and resolution to be recorded, as shown in Figure 2.3. OpenVDB, is another hierarchical tree data structure with efficient access methods to discretized volumetric data that has also been reported for robotic mapping applications [Besselmann et al., 2021].

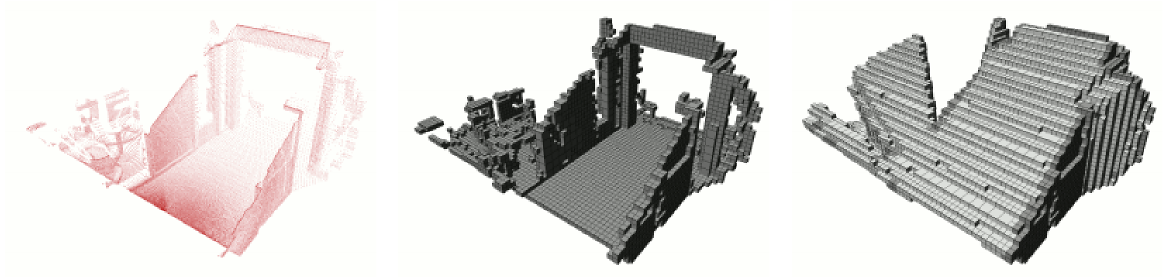


Figure 2.3: Given multiple point-clouds a full detailed map can be reconstructed using OctoMap. From left to right: a point cloud of the environment, the occupied cells and free cells [Hornung et al., 2013].

Other types of 3D representations can deal with obstacles directly, such as Voxblox, which uses Euclidean Signed Distance Fields (ESDFs) [Oleynikova et al., 2017]. The method proposed by Oleynikova et al. [2017] employs Truncated Signed Distance Field (TSDFs), initially designed for surface mesh reconstruction and common in computer graphics and vision, to incrementally generate ESDFs. The authors showed that in some cases, the proposed method can outperform OctoMap in speed. An example of a Voxblox-generated map can be seen in Figure 2.4. TSDFs are also used as a method for representing occupancy [Saulnier et al., 2020].

Hybrid approaches can also benefit from the use of three-dimensional topological

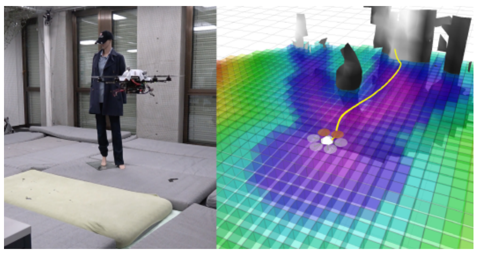


Figure 2.4: A Voxblox-generated map generated by a small UAV. The TSDF is shown as grayscale mesh, and the ESDF is shown as a single horizontal slice of the 3D grid [Oleynikova et al., 2017].

data. In Blochliger et al. [2018], a feature-based map generated from a visual SLAM system is converted into a 3D topological map using the Topomap algorithm. A topological map is created by extracting the occupancy information from the point cloud, and then a set of convex free-space clusters are estimated and used as the topological map's vertices (Figure 2.5).

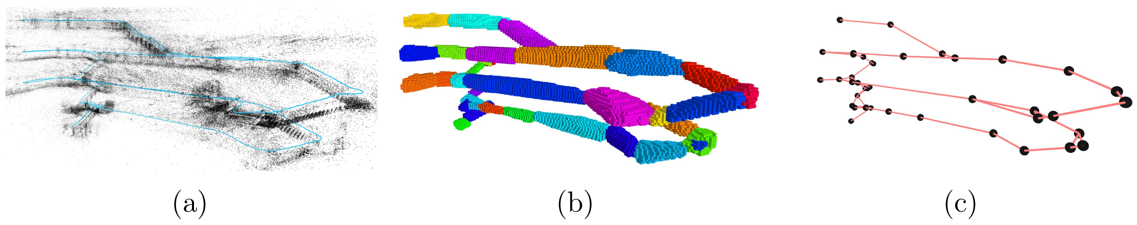


Figure 2.5: A topological map generated by point clouds: (a) sparse point cloud, (b) topological cluster of convex free space, and (c) the derived simplified navigation map between the clusters [Blochliger et al., 2018].

Another type of map representation with high description capabilities is a mesh. A mesh can be defined as a collection of vertices, edges, and faces that define a polyhedral object's shape. Generally, the faces of a mesh are triangles but may also be other geometrical figures. Compared to 2.5D solutions, 3D mesh surfaces allow for planning and navigation in arbitrary complex environments, including multi-level environments. Meshes can be used for navigation in terrestrial robotics and are useful for applications in uneven terrain [Pütz et al., 2016] (Figure 2.6).

Table 2.2 presents the characteristics of the most commonly used map representations and the expected usability in confined spaces. In this work, we use meshes, octrees, and point clouds. The meshes can represent with detail the roughness of the terrain, while the octrees and point clouds could be used to represent occupied and free space without the costly pre-processing required for the mesh reconstruction.

Table 2.2: Map representations characteristics and expected usability in confined spaces.

Map representation[*]	Dim.			Criteria					Remarks[‡]
	Two dimensional (2D)	Elevation based (2.5D)	Three dimensional (3D)	Low memory consumption	Low CPU overhead	Allows multi-level envs.	Multiple resolutions	Open source	
Feature/Line Maps [Sack and Burgard, 2004]	●	-	-	○	○	○	○	○	An intractable number of lines is necessary for a correct representation of complex terrain and obstacles.
Occupancy Grids [Moravec and Elfes, 1985]	●	-	-	◐	●	○	◐	●	Traditional occupancy grids are limited to 2D. Useful for obstacle avoidance, local traversability maps, or subterranean structured planar environments without multiple levels or challenging topography.
Topological Maps [Choset and Nagatani, 2001]	●	-	●	●	●	●	○	●	Can be used in conjunction of other types of maps, including 3D maps to improve path planning and semantic analysis.
Elevation Maps [Fankhauser and Hutter, 2018]	●	●	-	●	◐	○	◐	●	Cannot deal correctly with overhangs or multi level environments, *i.e.* areas with ceiling. Useful for local planning.
Point Clouds [Rusu and Cousins, 2011]	●	●	●	○	○	●	○	●	Great descriptive capabilities, not optimized for memory usage or fast search.
Meshes [Garland, 1999]	-	●	●	◐	◐	●	●	●	Capable of representing almost any shape. Memory-optimized. Can use textures. Surface extraction could be challenging with noisy datasets.
Voxblox [Oleynikova et al., 2017]	●	-	●	●	●	●	○	●	Novel description using Signed Distance Fields. Memory and space optimized.
Octrees [Hornung et al., 2013]	●	-	●	●	●	●	●	●	Great descriptive capabilities of almost any three dimensional shape. Allows multiple resolutions and its memory and space optimized.
OpenVDB [Besselmann et al., 2021]	●	-	●	●	●	●	●	◐	Hierarchical tree data structure with efficient access methods. Benchmarks shown that is faster than Octrees in some cases.

● = criterion fully covered; ○ = non-covered criterion; "-" = not applicable;
[*]Evaluated as standalone map representation;
[‡]Map combinations could overcome weak aspects of an individual representation, *i.e.*, point clouds with topological maps.

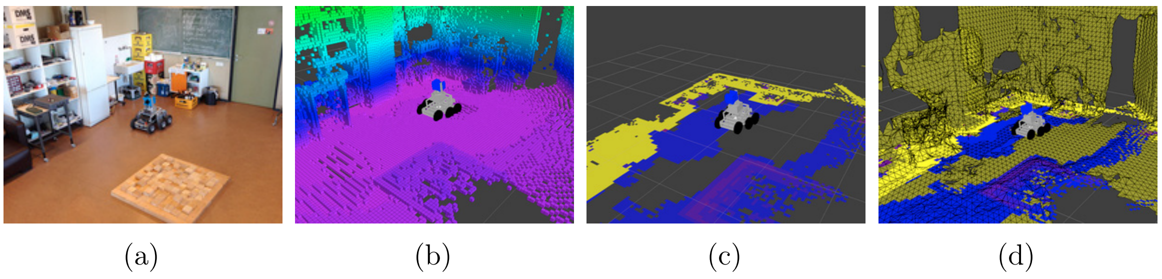


Figure 2.6: Map representations to estimate trafficability in 3D in uneven terrains: (a) real scenario, (b) Octree representation, (c) 2D map, and (d) mesh representation [Pütz et al., 2016].

2.2 Robot navigation in rugged terrains

A robot performing exploration tasks in rugged confined spaces will need to traverse complex 3D environments; therefore, traditional planning techniques that rely only upon 2D information are not adequate [Macenski et al., 2020]. Three-dimensional navigation was studied primarily for outdoor scenarios, including space exploration [Singh et al., 2000; Paz-Delgado et al., 2020]. Also, given the complexities and memory requirements for larger 3D maps, current navigation algorithms generally use a segmented approach to path planning via local and global maps. Global navigation mechanisms aim to guide the vehicle to its destination, while the local navigation cares about obstacle avoidance and other environmental fast changes.

As seen by many works in the area, graph representations allow increased flexibility in representing terrain topography and multi-level scenarios. Graphs are conceptual structures composed of nodes or vertices connected by arcs, where weights can be assigned to the arcs. Thus, a graph search algorithm can find the path with the lowest cost between two nodes. There are several graph-based search algorithms already used for robotic navigation [Karaman and Frazzoli, 2011a]. Approaches such as A* [Duchoň et al., 2014] or D* [Stentz et al., 1995] use heuristics to guide the search to an optimum solution in less time, given that the heuristic holds itself as admissible. A heuristic is admissible if it does not overestimate the cost of reaching the goal, *i.e.* the estimated cost is not higher than the lowest possible cost from the current point in the path. Other popular search algorithms, such as RRT [LaValle et al., 1998] and Optimum Rapidly-exploring Random Tree (RRT*) [Karaman and Frazzoli, 2011b], adopt a probabilistic strategy that generates a tree from a set of sampled vertexes. If computing efficiency is not a limitation, probabilistic or heuristical path generation algorithms may not be the most appropriate for path generation in complex 3D scenarios since slight deviations of the best path could cause robot failure. In those cases, deterministic optimal path search algorithms could be used, such as the Dijkstra Algorithm [Dijkstra, 1959].

In Ishigami et al. [2007], it is proposed a path planning algorithm to a planetary rover that considers the robot's dynamic mobility over 2.5D elevation maps. The authors use the Dijkstra's algorithm for path planning considering a cost function composed of terrain roughness, terrain inclination, and path length. These metrics are normalized and the trade-off between them is balanced trough weights. Thus, the algorithm evaluates the relevance of each one while finding navigable paths. In Schwarz and Behnke [2014] the authors presented a robot centered 2D drivability map generated from eight RGB-D sensors measuring the 3D geometry of the terrain around the robot using 2.5D egocentric height maps (Figure 2.7). Other works use a 2D projection of the drivable surface assuming that there are no overhanging structures the robot shall drive under or upon [Schadler et al., 2014]. The drivable surface is modeled as a graph consider traversability and drivability costs as edge and node costs. The optimal path is estimated via a graph search A* algorithm using the Euclidean distance as the heuristic.

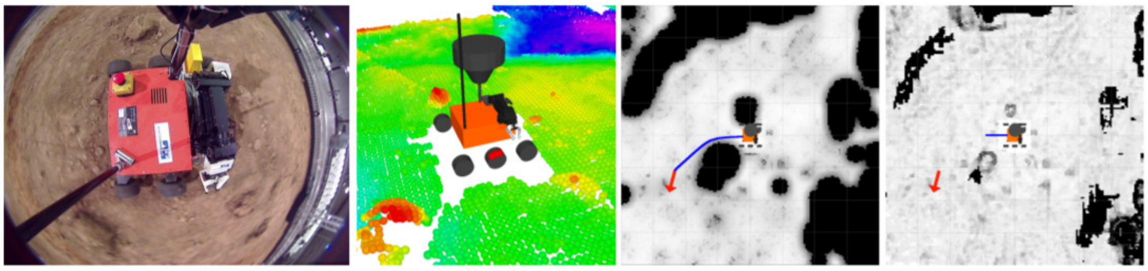


Figure 2.7: Navigation pipeline for rovers using multiple RGB-D sensors. From left to right: wide-angle overhead rover camera, point-cloud color-coded by height, drivability map and obstacle map [Schwarz and Behnke, 2014].

Raja et al. [2015] presents a motion planning algorithm for a six-wheel rover on rough terrain. The algorithm uses potential fields to represent the environment, where the proposed function consists of attractive, repulsive, tangential, and gradient forces. The gradient force is a function of the rover's roll, pitch, and yaw angles derived from the robot kinematic model. The algorithm tries to find safe paths avoiding routes with high gradient values. The authors attributed weights to the potential field function components, which are optimized using genetic algorithms. The proposed method also evaluates the vehicle's wheel velocity to ensure stability and prevent wheel slippage. In Jeddisaravi et al. [2016], the authors propose a multi-objective path planning for mobile robots for planetary exploration based on the A* path search. Given that extra-planetary surfaces could contain irregularities and hazardous situations such as holes, hills, and rocks, the method tries to minimize the difficulty, danger, elevation, and length of the path. The authors also used weights to set the trade-off between the objectives.

Modern approaches explore mesh structures for navigation, as shown in Figure 2.8 [Pütz et al., 2021]. By converting a mesh into a graph, the authors use the Dijkstra path search exploiting the mesh triangles' topological connections and their costs. Another planning step uses a wavefront propagation over the mesh surface to generate a potential field from each accessible position to the goal. Other approaches creates a watertight mesh for local navigation [Ruetz et al., 2019].

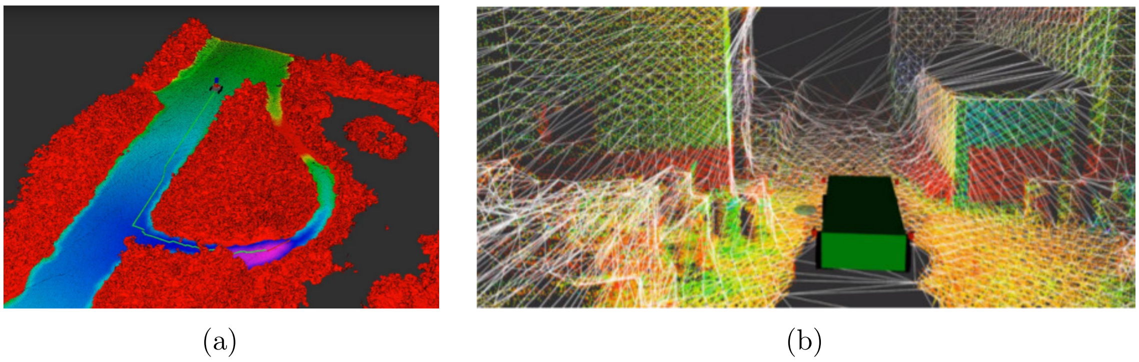


Figure 2.8: Mesh-based navigation examples. (a) Path planning instance showing wireframe rendering of a mesh and the visualization of vertex costs from red (higher cost) to green (lower cost) presented in Pütz et al. [2021]. (b) The robot-centric watertight mesh generated for local navigation proposed by Ruetz et al. [2019].

A complex point not explored heavily in recent navigation works is the interaction of a cable tether with the terrain as presented in Paton et al. [2020]. In this work, the authors focused on generating safe paths for rappeling rovers in complex terrains, considering the terrain-tether interactions, the stability and reachability constraints of the rappelling system. The ABIT* algorithm [Strub and Gammell, 2020] is used for path generation.

This work uses the optimal Dijkstra path planning algorithm and the RRT family of probabilistic path planners. The Dijkstra algorithm allows us to have a warranty that the best possible path was found, disregarding CPU consumption. The RRT family of algorithms is faster to compute than the deterministic Dijkstra, but it lacks that warranty of optimality; however since they are fast to compute, we use them to account for obstacles in dynamic environments.

2.3 Robotic exploration

As the act of robotic exploration means moving through an unknown environment while building a map, good exploration strategies should generate complete maps (or close to it, given the environment limitations) in the minimum time possible. Exploration is essential when dealing with coverage and mapping of an unknown region. The basic principles of exploration were described by Connolly [1985], which can be condensed as estimating the "next-best-views" in an unknown scenario.

The robotic exploration taxonomy can be divided into two large groups: two-dimensional and three-dimensional exploration. Those groups can also be subdivided into single or cooperative robots or classified by exploration strategy. The principal techniques used for exploration are frontier-based, sampling-based, information-based, and hybrid or random approaches. Figure 2.9 depicts a high-level taxonomy of typical robotic exploration approaches. Given the significant number of options and variants of the various exploration techniques, selecting the best algorithm depends upon the application at hand, the environment, and the available resources [Juliá et al., 2012].

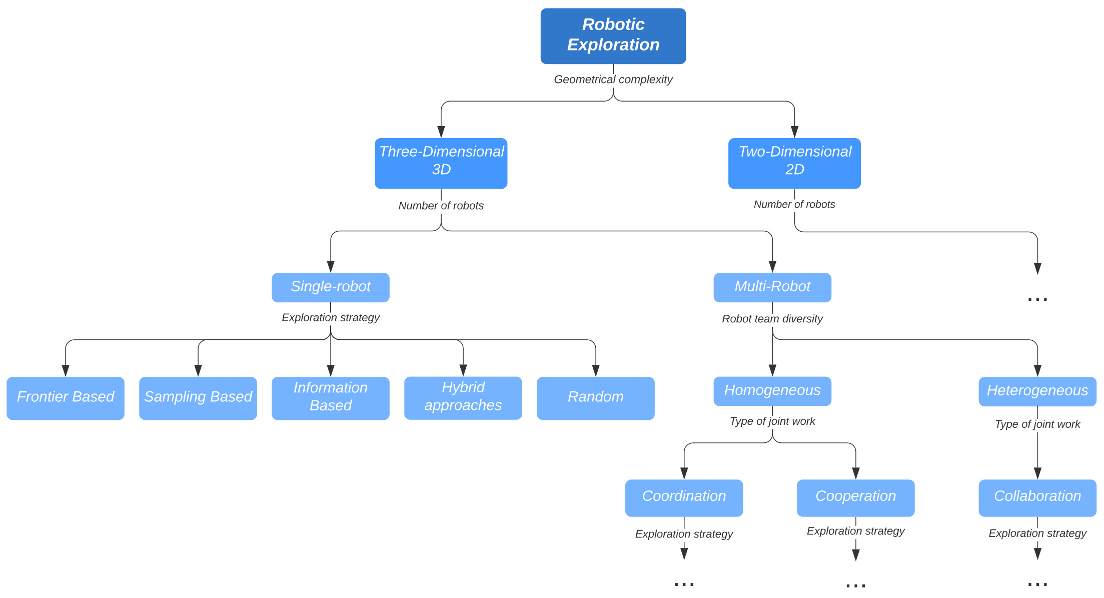


Figure 2.9: High-level robotic exploration taxonomy.

Two-dimensional exploration focuses primarily on structured indoor environments, such as buildings, offices, or cities. Buildings and other related structures are designed by humans (for humans) with standard and recognizable geometric features that some exploration algorithms can exploit. These approaches generally do not consider multiple levels or slopes and therefore are limited by more straightforward representations of the environment. This work focuses primarily on three-dimensional exploration since the terrain topography, obstacles, and multiple levels generally found on confined and subterranean environments require a more detailed understanding of the environment, unattainable with classic two-dimensional exploration approaches. Recent works have presented an in-depth analysis of multiple 2D exploration methods for readers unfamiliar with traditional exploration strategies [Lluvia et al., 2021; Sharma and Tiwari, 2016].

Three-dimensional exploration increases the complexity over traditional 2D exploration algorithms with the benefit of an increased range of real-world outdoor applications, ranging from drone inspection and reconstruction [Bircher et al., 2018] to cave mapping [Dang et al., 2020]. Performing exploration of an unknown 3D environment requires the acquisition of viewpoints over areas with unknown structures and textures in a problem commonly called the "next-best-view" selection [Connolly, 1985; Vasquez-Gomez et al., 2014].

A naive extension of 2D exploration methods into three dimensions introduces several challenges to overcome. For example, the popular frontier-based exploration method, first introduced by Yamauchi [Yamauchi, 1997] had as central idea the visitation of boundaries between unknown areas and the known open space. When used with raw, sparse, and noisy 3D sensors such as depth-cameras or 3D LiDAR, a naive frontier-based approach fails to accurately capture the differences between unoccupied and unknown space as those areas are often closely colocated [Shen et al., 2012]. Additionally, the sparse sensor information is easily mismatched for unknown free space, such as floor areas between LiDAR lines, yielding exploration strategies that drive a robot to produce a comprehensive local map at the cost of reducing the expansion rate of the global map.

A particular challenge for 3D exploration is computational complexity. Maintaining a dense map in two dimensions is computationally tractable; however, a 3D dense map can quickly become intractable if efficient mapping and exploration methods are not used. This limitation is critical for small mobile devices with limited payload and autonomy. In this sense, Shen et al. [2012] uses an efficient representation of the environment such as Octrees and performs exploration for single and multi-floor indoor scenarios with quadrotors. The proposed method shows a stochastic differential-equation-based exploration algorithm that only uses the known occupied space in the current map instead of the known free space and unknown space. In this method, the evolution of the stochastic differential equation simulates the expansion of a particle system with Newtonian dynamics, and the frontiers are detected as regions with more particle expansion. Other works have validated that efficient 3D next-best-view estimation can be achieved using Octrees and ray-tracing. In this idea, Vasquez-Gomez et al. [2013] proposes fast hierarchical sparse ray-tracing over partial Octrees. Multiple methods for next-best-view and information gain estimation, such as hierarchical and sparse ray-casting, are also benchmarked in Bissmarck et al. [2015].

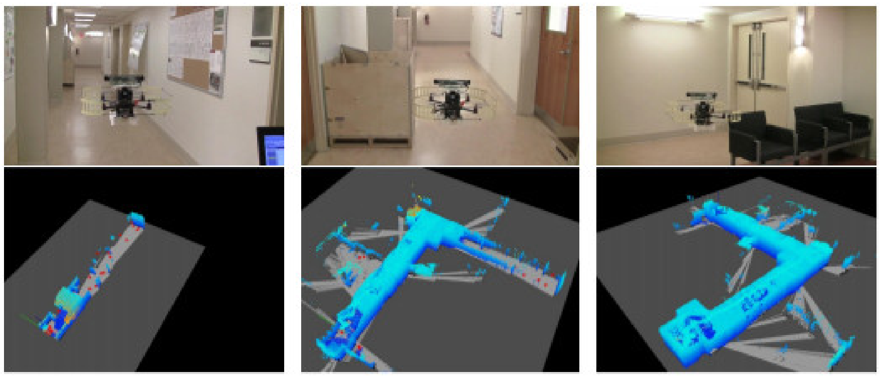


Figure 2.10: An example of the sequential 3D exploration of a single-floor hallway using a quadrotor and a depth camera. (Adapted from Shen et al. [2012]).

Many aerial exploration methods also rely on the free and known space extracted from an Octree to plan and perform decision making. Zu et al. Zhu et al. [2015] presented a 3D frontier-based exploration method named 3D-FBET using the state-changed space in a 3D map generated by Octrees. The method uses the transformation between two robots to find the correct transformation between multiple robots' map frames. By exploiting the probabilistic nature of Octrees, the resulting registered map is correctly aligned. In Wang et al. [2017], the authors present an exploration algorithm for 3D spaces with quadrotors that uses a potential information field to guide a robot toward a goal while being repelled by obstacles (Figure 2.11). They use a multi-objective function to select the best next frontier to visit considering distance and information gain. Then an Artificial Potential Field (APF) guides the robot to the goal. The proposed method only employs the goal region and local information around the robot, which reduces the computational complexity and can work online. The repulsive forces of repeatedly viewed voxels are increased to reduce local minima.

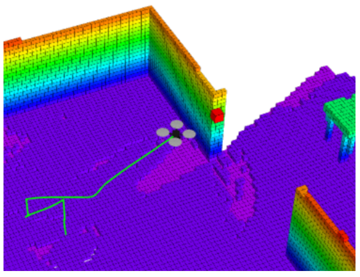


Figure 2.11: A UAV exploring indoor 3D spaces using an Octree representation [Wang et al., 2017].

In Cieslewski et al. [2017], the authors propose a method for quadrotors that allows fast exploration by maintaining the fastest speed possible while minimizing the angular movement and rotation of the robot at the first steps of the exploration procedure. The method generates instantaneous velocity commands based on the currently observed frontiers using an RGBD camera (dense range sensor) through an RRT* path-planning algorithm. If there are no more locally observable frontiers, the method falls back to traditional frontier exploration methods. An exploration method that performs online planning for quadrotors in a receding horizon fashion by sampling possible future configurations in a geometric random tree is proposed in Bircher et al. [2018] (Figure 2.12).

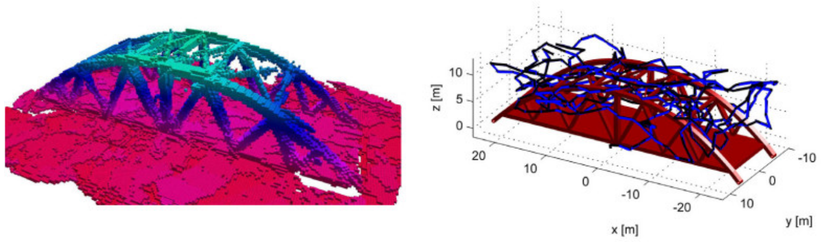


Figure 2.12: Resulting map after exploring a simulated 3D bridge with a quadrotor using the receding horizon next-best-view planner [Bircher et al., 2018]. The ground truth with the exploration path (solid blue lines) and the robot odometry (black lines) are on the right.

In Papachristos et al. [2019b], the authors present an uncertainty-aware path-planning strategy for autonomous aerial robots classified as active perception. The proposed method is a multistep procedure where the exploration gain is estimated, and an RRT* planner creates a geometric tree in the mapped free space. This mapping uses the camera sensor's field-of-view to estimate how much volume is mapped at the path points. The reduced uncertainty of reobserved voxels is also considered in the planning step. In a second step, possible deviations from the original path are planned, aiming at possible configurations that could yield improved localization or mapping confidence.

Dai et al. Dai et al. [2020] presented an algorithm for information-theoretic 3D exploration using UAVs. The method uses sparse ray-casting to estimate the information gain over the next-view candidates. A frontier utility is measured as the ratio between information gain (entropy) and the travel path time to reach a frontier. The feasible paths are generated by an informed RRT* algorithm exploiting an Octree's free known voxels. A sphere around the robot performs a collision check. Other works also use some kind of ray-tracing procedure over the view frustum of the robot to estimate the volumetric information gain of the frontiers [Papachristos et al., 2017; Delmerico et al., 2018; Witting et al., 2018].

Other types of robotic exploration, such as Degrees of Freedom (DoF) physical exploration, can also be performed using Information Theoretic approaches. Degrees of freedom exploration can be defined as the challenge of autonomously discovering and learning how to manipulate the environment by identifying promising points of interaction and pushing or pulling object parts to reveal new DoF and their properties. Opening drawers or doors, pushing a box, or pressing a button to switch on the light, can all be considered DoFs of the environment. In that direction, Otte et al. [2014] presents a method to decide where to explore based on the entropy reduction about the current robot's belief about the DoF. They propose a probabilistic belief representation to capture the robot's current knowledge state and a method to estimate the expected information gain from a set of new actions. Their method was validated in a real scenario with a PR2 robot, with motion planning and execution.

Many of the works on three-dimensional exploration are projected with aerial platforms in mind, given that terrestrial platforms are limited by the terrain topography, but novel platforms such as quadruped and multi-legged robots are gaining traction as capable exploration devices. In this context, Prágr et al. [2019] presented a method for exploration with motion efficiency using multi-legged robots such as a small hexapod platform. Their proposal uses the terrain traversal cost to estimate the best next goal. The traversal cost is estimated using the technique presented in [Prágr et al., 2018], extended by the use of a robust bayesian committee machine (RBCM) inference mechanism with Gaussian Processes (GP) experts. The employed strategy greedily improves the traversal cost by navigating terrain that is considered unknown. The robot explores the spatial frontiers if the observed terrain is sufficiently known. This method allows the estimation of the variance for the knowledge about the terrain, guiding the platform to areas where knowledge can be improved. Dijkstra's algorithm is used for path planning for the nearest frontier. Bayer and Faigl [2019] also presents an accuracy analysis on terrain mapping with the same platform and Intel Realsense cameras (T265 and D435). In this mapping analysis, the height is estimated utilizing a Kalman filter with the sensor model on the cell heights. A threshold determines a traversable path on the height difference among neighbors of a cell. The experiments performed with the platform show that the T265 camera provides localization with a similar absolute error as the ORBSLAM2 Mur-Artal and Tardós [2017] combined with D435 RGB-D camera but with less processing overhead.

Other 3D terrestrial exploration works use potential fields to navigate rugged terrain. In Maffei et al. [2020], the authors propose modeling exploration as a Boundary-Value Problem (BVP) for uneven 3D terrain. The solution is computed over a 2D grid associated with an elevation map, and the robot can explore the environment while

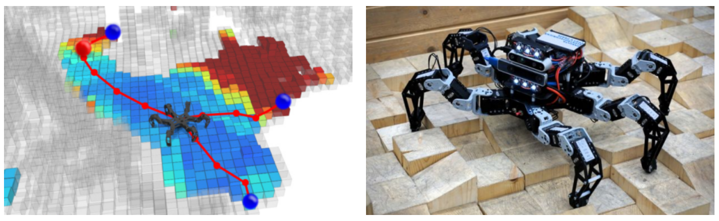


Figure 2.13: Terrain traversal exploration with a hexapod robot: (a) visualization of reasoning about possible navigational goals in the spatial frontier-based (blue spheres), and (b) real experiment in a controlled scenario [Prágr et al., 2019].

avoiding obstacles or other dangerous areas by following the gradient descent of the potential field.

2.4 Subterranean exploration using terrestrial platforms

One of the first works in subterranean exploration using autonomous robots is presented in Thrun et al. [2004]. In this proposal, the authors show the inspection of an abandoned mine with a 1,500-pound custom-built terrestrial robot. Considering that the environment was flat and the robot had considerable size, traditional 2D navigation techniques allowed the robot to successfully transit the environment; nevertheless, most recent works consider a more detailed 3D environment representation to address the complexities of multi-level caves and overhang obstacles.

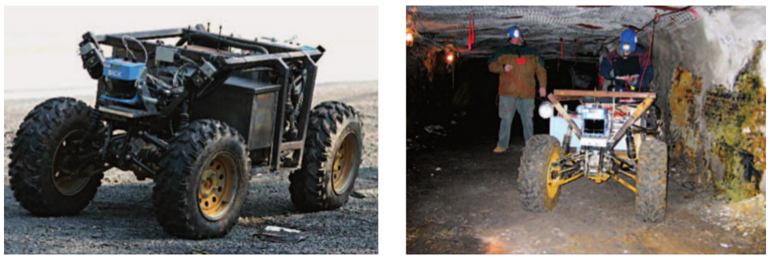


Figure 2.14: Autonomous exploration in abandoned mines [Thrun et al., 2004]. On the left the Groundhog robot used for autonomous exploration is depicted: a 1,500-pound custom-built vehicle equipped with onboard computing, laser range sensing, gas and sinkage sensors, and video recording equipment. On the right, the equipment entering the Bruceton Research Mine.

Terrestrial robots, being wheeled, tracked, or legged platforms, have considerable payload and battery autonomy advantages over aerial platforms, even though path planning and navigation is still a challenge for these types of devices on rugged and multi-level terrains. Recent hardware advances allow the purchase of commercial off-the-shelf quadruped robots. Although quadruped robots have particular locomotion complexities, these platforms are versatile, surpassing other terrestrial counterparts in many scenarios, such as climbing stairs or obstacle negotiation Fankhauser and Hutter [2018], and have sparked interest in novel locomotion, sensing and navigation methods applicable in confined spaces [Buchanan et al., 2019; Wisth et al., 2021; Buchanan et al., 2021].

In Ebadi et al. [2020] the authors present the Large-scale Autonomous Mapping and Positioning (LAMP) system, a LIDAR-based SLAM system for multiple terrestrial robots developed in the context of the DARPA Subterranean Challenge. The cooperative module uses a centralized base station that receives each robot's pose graphs within communication range and merges them into a shared pose graph, performing loop closures as needed. Experiments were performed in a cave environment with a team of wheeled ground robots (Husky A200). In an extension of previous works in 2.5D navigation, Bayer [2020] presented a faster exploration framework for ground robots with a hybrid uniform grid-based and tree-based map representation. The tree structure enables fast access to neighboring cells and a low memory footprint. The frontier with the lowest traversable cost is selected for visitation using an iteration-bounded A* algorithm.

In Ohradzansky et al. [2020], the authors propose a reactive bio-inspired exploration technique for corridor-like environments that uses a metric-topological graph map. In this proposal, horizontal depth scans generate a centering response for graph navigation, inspired by the optical flow present in various insect's visuomotor systems. The topological map is generated using image processing, and the robot iteratively explores the nearest unvisited node in the graph.

Legged and quadrupedal robots are gaining traction thanks to their extended mobility capabilities over traditional wheeled robots that improve terrain traversability and obstacle bypassing while maintaining a reasonable payload capacity, size, and endurance. In this vein, Bouman et al. [2020] presents an autonomy framework that enables mapping, odometry, navigation, and information-theoretic exploration with legged robots such as the Spot Mini from Boston Robotics (Figure 2.15). Traversability maps are generated considering slopes and positive and negative obstacles. A cooperative quadruped exploration team is presented in Miller et al. [2020], showing a distributed database mesh networking system for interrobot communication. This work performs path planning by discretizing the LiDAR scan into a heightmap, and filtering out the ceiling. After estimating metrics such as the gradient of the map, rotation cost, distance cost, traversability cost, sidestep cost, and reversal cost, Dijkstra's algorithm is used over the configuration space to estimate the best path. The robots explore different areas of the environment using a set of initial commands from the operator, and the exploration behavior is frontier-based, leading a robot to the frontier closest to its current heading.

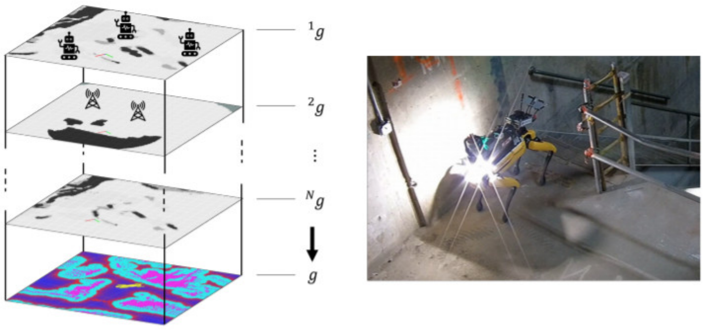


Figure 2.15: A multi-layer traversability map for quadruped robots on the left and the Spot Mini robot during an exploration task on the right [Bouman et al., 2020].

2.5 Contextualization of this work

The pipeline presented in this work for autonomous explorations differs from most works by having a path planner that does not rely on 2D or 2.5D maps for safe navigation. Most works assume that the 3D map can be converted into a 2D or 2.5D map, which is not always the case when dealing with rugged and confined spaces, where the roof is not always predictable, with the presence of overhangs or other vertical structures. The current proposal also estimates the best-next frontier by using an information-theoretic volumetric analysis novel for 3D environments with sparse LiDAR data, as most exploring algorithms use dense 3D data acquired from other types of sensors such as RGBD or ToF cameras. We also presented a novel biased RRT algorithm, called MI-RRT, that simplifies the exploration procedure as it can select and plan to frontier candidates simultaneously. Our current local planner uses the visibility of point clouds from the robot's Point of View (PoV) to determine which places can be locally reachable and detect dead-roads or obstacles in dynamic environments.

In contrast to the state-of-the-art, the proposed sensor suite in this work is minimal: one 3D LiDAR, IMU, and wheel odometry. Despite this, we can perform the same complex exploration mission with fewer sensors, increasing autonomy and decreasing weight.

As presented in the previous sections, exploratory missions for ground robots considering terrain traversability is a challenging novel research area, primarily led today by research in legged robotics and the advances presented in the DARPA's subterranean challenge, and there are still many open research questions. A summarization of the exploration works that are more related to our proposal is shown in Table 2.3.

Table 2.3: Summary of the principal related works for exploration in confined and subterranean scenarios. Proposed methodology is highlighted in the last row.

Work ref.	Exploration and autonomy criteria									Comments
	Multi-robot	Aerial	Terrestrial	Confined spaces	Dynamic environment	Two dim. (2D)	Elevation (2.5D)	Three dim. (3D)	Map type	Unreliable comms	
Related works (sorted by year/author)											
Thrun et al. [2004]	○	○	●	●	○	●	●	○	Gridmap	●	Subterranean mine, 2D, pioneer work in the area.
Dang et al. [2019]	○	●	○	●	○	○	○	●	Octomap	●	Graph/information based. RRT*. Dijkstra. GBplanner.
Papachristos et al. [2019a]	○	●	○	●	○	○	○	●	Octomap	●	Graph-based. Receding Horizon. MPC.
Qin et al. [2019]	●	●	●	●	●	●	●	●	Octomap	◐	Cooperative information/frontier based.
Akbari et al. [2020]	○	●	○	●	○	○	○	●	Octomap	◐	Semantic graph/frontier based. Ray casting.
Bayer [2020]	○	○	●	●	○	○	●	○	Quadtree	○	Grid/Tree based exploration.
Bouman et al. [2020]	●	○	●	●	●	○	●	○	Voxblox/MLT	●	Quadrupeds. Graph/information based exploration.
Dharmadhikari et al. [2020]	○	●	○	●	○	○	○	●	Voxblox	●	Motion primitives/information based.
Ebadi et al. [2020]	●	○	●	●	●	○	●	○	Pointcloud	●	Graph-based cooperative SLAM.
Miller et al. [2020]	●	○	●	●	●	●	●	○	Height map	●	Frontier-based. Distributed mesh networking.
Ohradzansky et al. [2020]	○	○	●	●	○	●	○	○	Topological	○	Bio-inspired, metric-topological LiDAR map.
Agha et al. [2021]	●	●	●	●	●	●	●	●	Gridmap/Height map/Pointcloud	●	Graph/information based exploration. Autonomy pipeline. Breadcrumbs comms
Ahmad et al. [2021]	○	●	○	●	●	○	○	●	Voxblox	●	Frontier-based exploration. Obstacle reactive.
Current methodology											
Azpúrua et al.[‡]	○	○	●	●	●	○	○	●	Mesh/Octomap/Pointcloud	●	Graph-based, information-based, obstacle avoidance with local/global MI-RRT. Uses meshes, Octomap and point clouds.

● = criterion fully covered; ◐ = partially covered criterion; ○ = non-covered criterion;
‡ The methodology proposed in this dissertation.

Chapter 3

Navigation in rugged terrains

"Plans are nothing; planning is everything."

— Dwight D. Eisenhower.

Autonomous mobile robots require adequate reference paths capable of being performed by the platform's underline control and localization system to accomplish any task that involves locomotion. In this sense, the following chapter proposes a functional approach for path planning in rugged and confined scenarios that consider the terrain topography to generate the traversable paths; we answer the question: "How the robot reaches the goal?" (Figure 3.1). The proposed approach generates terrain-aware reference paths based on a mesh reconstruction pipeline tailored for confined spaces. By modeling the environment's reconstructed mesh as a graph, optimal search algorithms can optimize a map given one or multiple terrain metrics. A critical point for path planners is representing the robot's contact points with the ground, which are complex to reproduce at the planning phase. In this regard, we also propose multiple methods of realistic robot pose estimation with the reconstructed ground topography.

Figure 3.1: Chapter question: "How the robot reaches the goal?".

3.1 Problem Formulation

In this chapter we address the problem of generating safe paths for terrestrial mobile robots in rugged terrains at complex three dimensional environments. The task will be executed by a single robot R, where its pose q is represented by a configuration $\mathbf{q}_\mathrm{k} \in \mathrm{SE}(3)$.

The robot needs to traverse a static environment $\mathcal{E} \in \mathbb{R}^3$, which poses critical challenges for the navigation, for example, obstacles, uneven terrain, and narrow passages.

Let $\mathcal{M}$ be a three-dimensional occupancy grid representation of $\mathcal{E}$, generated by the observations of a 3D range sensor. Given a map $\mathcal{M}$, an initial reachable position $n_i \in \mathcal{M}$, and a goal position n_{goal}, the robot must define a safe and efficient continuous path $p = \{n_1,\ n_2,\ ...,\ n_n,\} \to \mathcal{M}$, such that the maximum slope angle traversable by the robot (θ_{max}) is respected, $p_0 = n_i$ and $p_n = n_{goal}$. The map $\mathcal{M}$ can have discontinuities, multi-level floors, slopes and other typical characteristics of subterranean confined spaces.

Problem 1 (Three-dimensional navigation in rugged terrains)**.** *Given a ground robot R in a rugged static environment $\mathcal{E}$. The problem consists of efficiently generate a safe path p using the known occupied cells in $\mathcal{M}$ given a start location $n_i \in \mathcal{M}$ and target location n_{goal}. For that, we must:*

- *Filter the untraversable and unreachable areas of the environment given the locomotion limitation of the platform (θ_{max}).*
- *Estimate the pose q of the robot at a given point in $\mathcal{M}$ using the robot physical characteristics.*
- *Define terrain-aware metrics capable of modeling the hazards and challenges of rugged terrains.*
- *Generate an optimal path p from n_i to n_{goal} that is safe and efficient.*

3.2 Proposed Method

Natural caves commonly present rough terrain, which is challenging for path planning. Complex landscapes require a complete three-dimensional map for planning a safe and efficient robotic locomotion. As shown in Chapter 2, three-dimensional meshes are adequate for this task since they can represent any 3D shape, generating highly descriptive environment models. Unlike 3D point clouds, meshes have the face area, position, and normal information, which help terrestrial robot mobility. Other representations such as elevation maps have difficulties representing some cave structures, e.g., arcs, narrow tunnels, or terrain overhangs. In that regard, we extend the path planning pipeline presented in Santos et al. [2018] and Azpurua et al. [2019] by integrating an automatic mesh generation process inside the planner, including a pre-processing step for filtering non-reachable regions and a fast method to compute the interaction of the robot with the ground. In contrast to our previous works in path planning, this pipeline is fast enough to be executed online at the exploration and mapping phases directly on the robot's onboard computer. The proposal is also fully integrated with a low-level control algorithm based on Vector Fields. Figure 3.2 shows the proposed path planning pipeline.

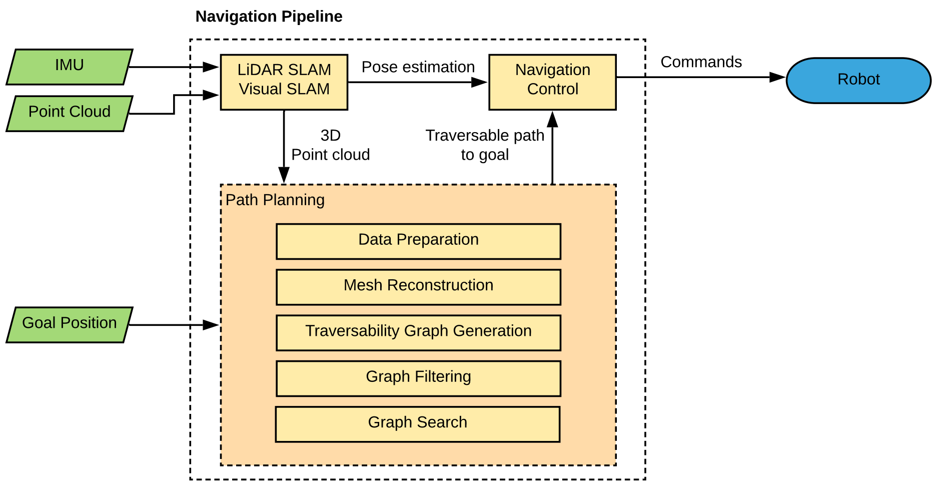


Figure 3.2: General path planning and navigation workflow.

3.2.1 LiDAR SLAM for Confined Spaces

Closed and confined spaces are challenging for most mapping methods. Even state-of-the-art Visual-SLAM methods such as the Real-Time Appearance-Based Mapping (RTAB-Map) algorithm [Labbé and Michaud, 2019] may be impaired by poor lighting conditions, which difficult the perception of visual features with monocular or depth cameras. In this sense, a suitable solution that is less dependant on environmental conditions is LiDAR-based SLAM. As shown in Rezende et al. [2020], LiDAR sensors can provide robust measures of distances from the objects in the environment regardless of lighting conditions.

For the LiDAR-SLAM, we use the Lightweight and Ground-Optimized Lidar Odometry And Mapping (LeGO-LOAM) methodology proposed by Shan and Englot [2018], which is a light-weight version of the LOAM technique, optimized for land vehicles since it assumes there is always a ground plane in the scan as captured via a multi-line LiDAR sensor. The LeGO-LOAM is an efficient mapping method, capable of good performance even in the limited embedded robot's computer. A high-level diagram of the LeGO-LOAM algorithm is described in Figure 3.3.

This method was validated by da Cruz Júnior et al. [2020] for indoor, closed spaces and underground caves, thus presenting itself as a practical mapping method for confined spaces.

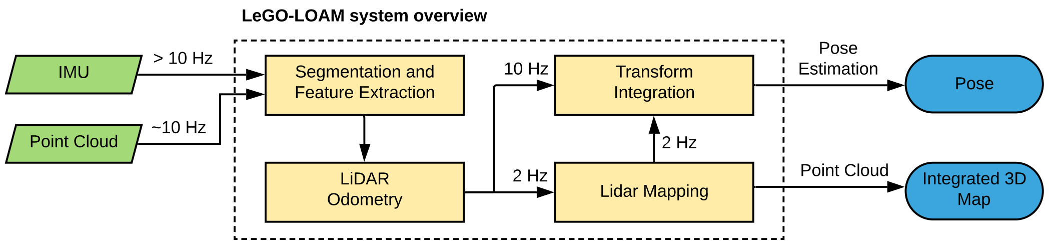


Figure 3.3: LeGO-LOAM SLAM system overview, adapted from Shan and Englot [2018] and Azpúrua et al. [2021b].

3.2.2 Mesh Reconstruction

Meshes are three-dimensional structures capable of high-dimensional accuracy representation. Mesh objects are generally a well-behaved set of joined faces without empty spaces between vertexes, in contrast to traditional point clouds. Therefore, in this work, we need to convert a point cloud into a coherent mesh object to estimate terrain-aware paths. Mesh reconstruction is generally an error-prone process that needs considerable parameter tuning and consumes a significant amount of CPU resources. In this sense, most mesh reconstruction techniques are not suited for online reconstruction, especially from noisy robot-generated data. We assume that for a reconstruction algorithm to be considered online-capable, it should generate meshes in a reasonable amount of time to allow the successful realization of a task. The notion of an appropriate delay varies from task to task; however, we assume that a range from a couple of seconds to a few minutes is a reasonable amount of time for planning exploratory missions in static environments.

Automatic reconstruction of cave-like environments is a particular challenge for reconstruction algorithms since the algorithm must preserve holes and other delicate components of the environment to generate a faithful three-dimensional mesh model.

Noise and misalignments are common in point clouds generated by mobile robots; therefore, a reconstruction method must tolerate these types of errors. In this regard, although an experienced 3D expert can produce high-quality meshes using advanced software pipelines, it comes at the price of spending plenty of time tweaking a large number of parameters, which is not practical for many real-world situations.

In this work we adapted the *Surface Recon*[*] mesh reconstruction algorithm proposed by Potje *et al.* at the VeRLab laboratory[†], which provides an intuitive set of parameters available for non-expert users to adjust and, in most cases, works out-of-the-box. These characteristics made *Surface Recon* an adequate proposal for an online and automated mesh generation pipeline. The adapted algorithm has three main steps: (i) Normal estimation, (ii) Surface reconstruction, and (iii) hole filling. The color embedding methods from the original algorithm were removed from the pipeline to speed up the process since we are dealing with LiDAR data without any texture attached. Figure 3.4 illustrates the mesh generation pipeline.

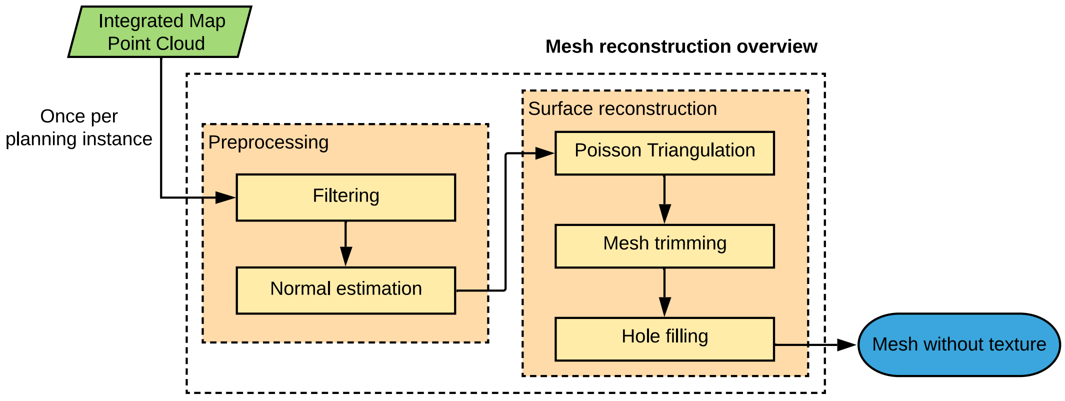


Figure 3.4: Mesh reconstruction pipeline overview, adapted from Azpúrua et al. [2021b].

At the preprocessing step, we filter the map point cloud generated by the LiDAR-SLAM algorithm to reduce noise and estimate a reduced set of points representing the environment faithfully. Since many real point clouds generated by the robot have a great deal of noise and redundancy (close points), we perform the following initial filtering steps: eliminating overlapping points using a KD-tree algorithm and the size reduction of large clouds using uniform random sampling. A consistent normal estimation is a critical step for our path planning approach since a flipped or erroneous normal could render a perfectly traversable node as an obstacle. The face normal estimation process involves fitting a local plane using the singular value decomposition (SVD) of the covariance matrix given the K-nearest neighbors of a point. A Riemannian graph

[*]`www.github.com/verlab/mesh-vr-reconstruction-and-view`
[†]`www.verlab.dcc.ufmg.br`

from the K-nearest neighbors is generated for all the points, encoding the neighboring tangent plane centers' geometric proximity. The normals are oriented by generating a minimum spanning tree (MST) from over the robot's position instead of the highest point in the Riemannian graph [Hoppe et al., 1992] to guarantee that outliers from the point cloud do not result in reconstruction problems such as partial reconstructions or flipped normals.

The point sets with the estimated oriented normals are the input for a Poisson Surface Reconstruction algorithm Kazhdan and Hoppe [2013] which generates an implicit function. This implicit function is an approximate indicator function of the inferred solid being reconstructed. The surface mesh is finally generated by extracting an isosurface of this function using the method proposed by Rineau and Yvinec [2007]. The vertex and faces are trimmed from the mesh if the separation to their closest neighbors is greater than the average separation of the original cloud points to reduce noise generated by the Poisson reconstruction.

At the final steps, the holes are filled in the trimmed mesh using the technique presented by Liepa [2003], with a threshold based on the average point distance to prevent cave entrances and other passages from being filled. The sequential process of mesh generation is depicted in Figure 3.5.

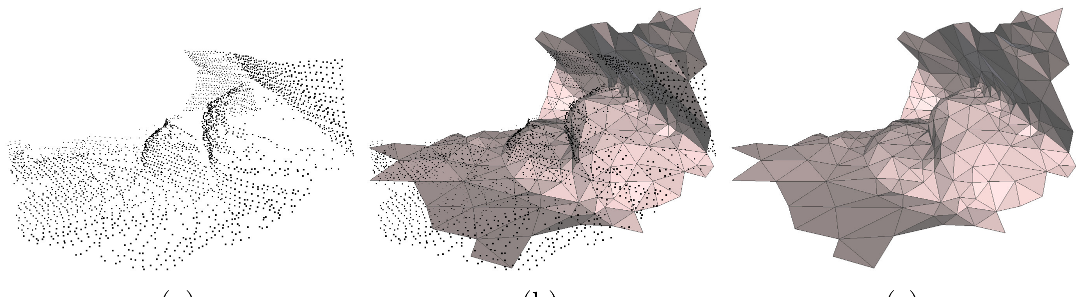

(a) (b) (c)

Figure 3.5: Example process for converting a raw point cloud of the environment into a mesh: (a) input point cloud segment of a cave map, (b) overlay of the points over the estimated reconstruction and (c) complete mesh model.

3.2.3 Path planning

After defining the next target point where the robot should move to explore the uneven environment (n_{goal}), it is necessary to compute feasible paths connecting the robot's current position and defined goal.

The mesh structure, a collection of vertices, edges, and faces, helps path planning in rugged terrains since it facilitates the analysis of some terrain characteristics such as topography and inclination. One of these structural benefits is the availability of the mesh face's normal vector. The normal vector of a face ($\vec{\mathbf{n}}^z$) –generally just called "normal"–, is a vector that is perpendicular to the surface center point, meaning that the normal is a cue that points us the inclination direction. This inclination value is primarily used for pruning and terrain analysis. A mesh with estimated normals is depicted in Figure 3.6.

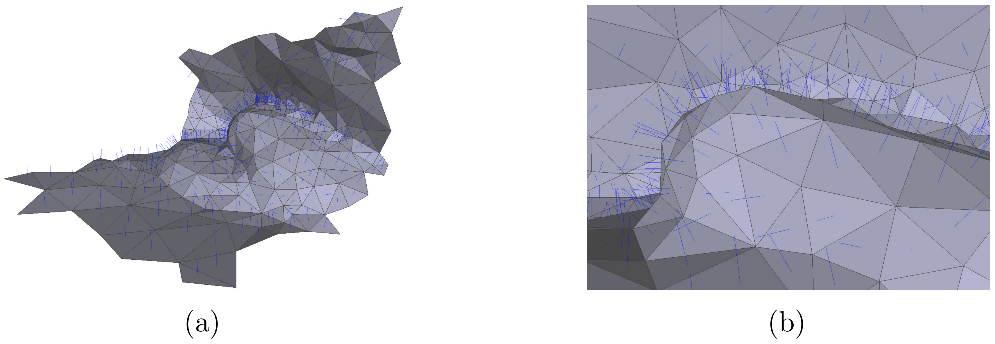


Figure 3.6: Triangle mesh of a cave environment: (a) mesh with z normal vectors ($\vec{\mathbf{n}}^z$) depicted in blue, and (b) a zoom-in to the mesh faces.

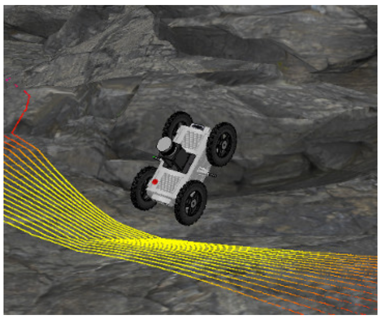
The path planning pipeline models the 3D mesh representing the environment as a graph $\mathcal{G} = (V, E)$, where the vertices V are the centroids of the mesh faces, and the edges E connect neighboring faces with an associated cost regarding the adopted terrain traversability metric. A node's neighbors are all other nodes that share a face or mesh vertices with the current node. Over this graph, we use the well-established Dijkstra algorithm to calculate optimal paths according to different metrics, such as distance, traversability, energy consumption, or a combination of them.

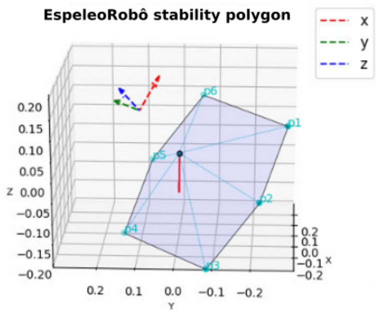
The selection of Dijkstra's algorithm over other probabilistic search methods is justified by the fact that probabilistic approaches could generate nonoptimal solutions. The proposed method uses an optimal path search to generate the best possible outcome regardless of CPU consumption. Nevertheless, heuristic approaches were also validated in Chapter 5 to reduce the path estimation processing overhead, particularly when using the more CPU-intensive terrain interaction methods.

The traversability graph generation $\mathcal{G}$ used for planning involves the following steps:

Removing untraversable areas Estimating reachable and non-reachable regions is particularly crucial for ground robots performing exploration tasks autonomously in uneven terrains. To generate a traversability map, we iteratively transform the point cloud of the environment into a mesh $\mathbb{M}$ to estimate slopes. The slopes are estimated by extracting the z normal vector ($\vec{\mathbf{n}}^z$) for every mesh face. The mesh has a collection of faces $\mathbb{F}$, and it is filtered considering the maximum slope angle traversable by the robot (θ_{max}).

The maximum slope angle traversable by the robot θ_{max} could be estimated by the techniques proposed by Messuri and Klein [1985]; Freitas et al. [2010]; Rocha et al. [2019], where a support polygon is generated to represent the interaction the robot has with the terrain (Figure 3.7). This support polygon $\mathcal{SP}$ is estimated as the convex hull of the robot's terrain contact points, indicating where the wheels or legs touch the ground. Many stability metrics could be estimated with this approach; one of the more interesting is the projection of the robot's center of gravity given a rotation angle. The center of gravity projection helps detect instabilities when the center of gravity is estimated outside the support polygon's borders for a given robot pose. In this work, we assume θ_{max} is given.

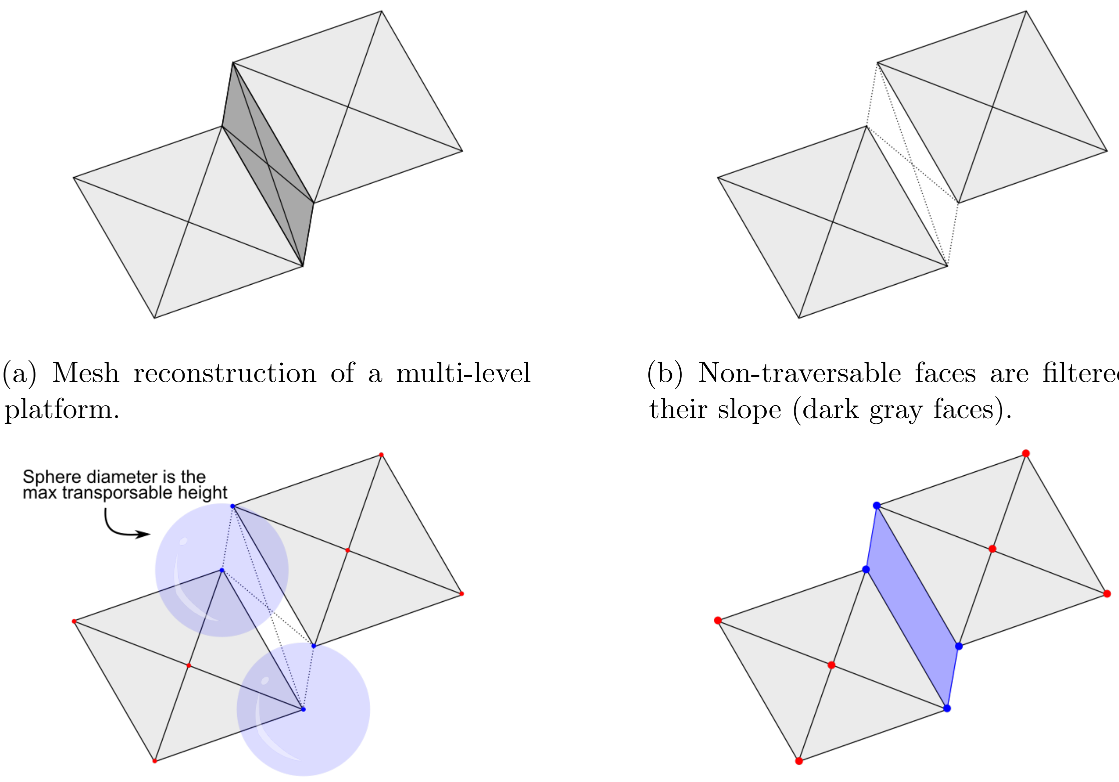

(a) Robot's real pose.



(b) Support polygon representation.

Figure 3.7: Robot stability analysis using the support polygon: (a) the real pose of the robot, and (b) the support polygon of the robot contact points representing the robot's pose.

Joining unconnected traversable areas In industrial scenarios, it is expected the presence of multi-level areas separated by a steep slope, such as stairs or similar structures. Even rocks or other obstacles could be removed from $\mathcal{G}$ by their steep slope, but a vital connection between the known map and an interesting unexplored area could also be filtered with it. Sometimes, the distance between traversable areas is small enough so that the robot is able to bypass it given the wheel type, radius, and other robot characteristics. In this sense, the remaining unconnected traversable stages separated by the initial slope filter can be connected via a *bumpiness* threshold τ_{bump}, calculated from a 3D sphere with a fixed radius from the closest vertices between the sub-graphs using a KD-tree search. The τ_{bump} threshold is a conservative value that might be estimated and assigned via experimental evaluation of obstacle bypassing by the robot platform; in this work, we assume this threshold value was already estimated. The process of joining traversable unconnected areas is depicted in Figure 3.8.



(a) Mesh reconstruction of a multi-level platform.

(b) Non-traversable faces are filtered by their slope (dark gray faces).

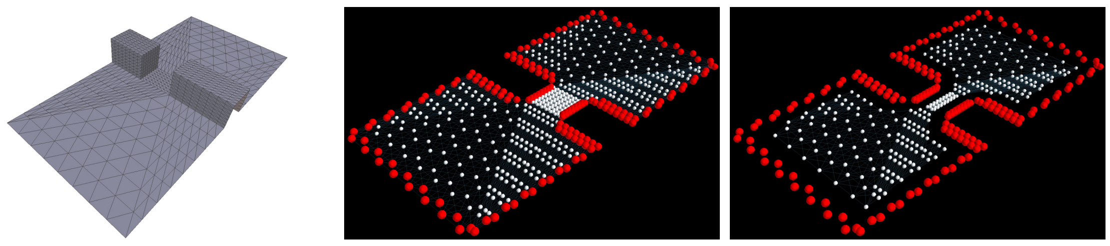
(c) A sphere is used to search nearby unconnected areas.

(d) The final graph considers the joined platforms as traversable.

Figure 3.8: Iterative process of estimating the bumpiness over the $\mathbb{M}$ and $\mathcal{G}$ using a sphere of diameter τ_{bump} to join reachable non-connected platforms.

Border inflation Often, the best paths that minimize the total distance are too close to obstacles or other dangerous map areas. A simple way to remove spaces too tight for the robot to go safely is performing border inflation. Border inflation allows us to expand the border limits inwards the graph, reducing the traversable space the robot has available to plan. In this sense, we define a threshold radius, $\tau_{inflation}$ that should be the minimum radius that circumscribes the polygon describing the robot footprint $\mathcal{SP}$. We apply a radius of $\tau_{inflation}$ at every border node in $\mathcal{G}$ and remove the nodes and edges inside the radius. The process of border inflation over a simple scenario with two obstacles can be observed in Figure 3.9.

Planning algorithm Algorithm 1 describes the process of generating the traversable graph G and final path p, where $\mathbb{F}$ is the set of mesh's faces, $\hat{\mathbb{M}}$ is a mesh $\mathbb{M}$ without



(a) Original mesh. (b) Traversability graph. (c) Expanded borders.

Figure 3.9: Iterative process of inflate borders in $\mathcal{G}$, using the $\tau_{inflation}$ threshold.

the faces considered untraversable, n_{start} is the position of the robot and n_{goal} the desired goal node. The resulting path $p = \{n_1,\ n_2,\ ...,\ n_i,\ ...,\ n_{goal}\}$ will consist of neighboring traversable nodes and $\bar{p} = p \backslash n_1$.

Algorithm 1: Exact path planning generation ($\mathcal{M}$)

```
$\mathbb{M} \leftarrow generateMesh(\mathcal{M})$
$\mathbb{F} \leftarrow \mathbb{M}_{faces}$
for $i \leftarrow 1$ to $|\mathbb{F}|$ do
    if $\vec{\mathbf{n}}_i^z < \theta_{max}$ then
        $\hat{\mathbb{M}} \leftarrow \mathbb{M} \setminus \{\mathbb{F}_i\}$        ▷ Remove face
    end
end
$\mathcal{G} \leftarrow graphFromFaceCentroids(\hat{\mathbb{M}})$
$\mathcal{G} \leftarrow KDTreeTraversablePlatformConnect(\mathcal{G}, \tau_{bump})$
$\mathcal{G} \leftarrow removeNonConnectedComponents(\mathcal{G})$
$\mathcal{G} \leftarrow inflateBorders(\mathcal{G}, \tau_{inflation})$
$\mathcal{G}_w \leftarrow estimateEdgeWeights(\mathcal{G})$    ▷ Estimate terrain-aware edge weights
$p \leftarrow DijkstraSearch(\mathcal{G}, \mathcal{G}_w, n_{start}, n_{goal})$
return $p$
```

3.2.4 Terrain-aware metrics

The terrain-aware metrics such as traveled distance, traversability and energy consumption used as the edges weights E in G, are computed with the approaches described in the following sub-sections.

3.2.4.1 Traveled Distance Metric

This metric aims to find the shortest path from the robot's current start position to a goal. The cost function using this metric is defined as follows:

$$C_1(p) = \sum_{n \in \bar{p}} D(n), \tag{3.1}$$

where the route $p = \{n_1,\ n_2,\ ...,\ n_i,\ ...,\ n_{goal}\}$ consists of a set of neighboring nodes, $\bar{p} = p \backslash n_1$, and $D(n)$ is the 3D euclidean distance between centroid of the faces representing the nodes n and $n-1$ (Figure 3.10). Consider n_1 as the current node. This metric does not considers the terrain topography, and if used alone it usually generates very dangerous shortest paths for the robot to follow.

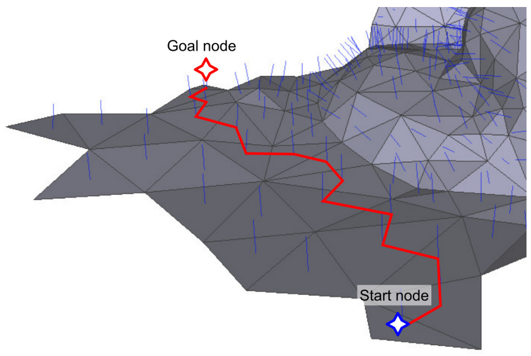


Figure 3.10: Example of a path using the shortest distance, estimated from the centroid of the mesh face represented by nodes from $\mathcal{G}$.

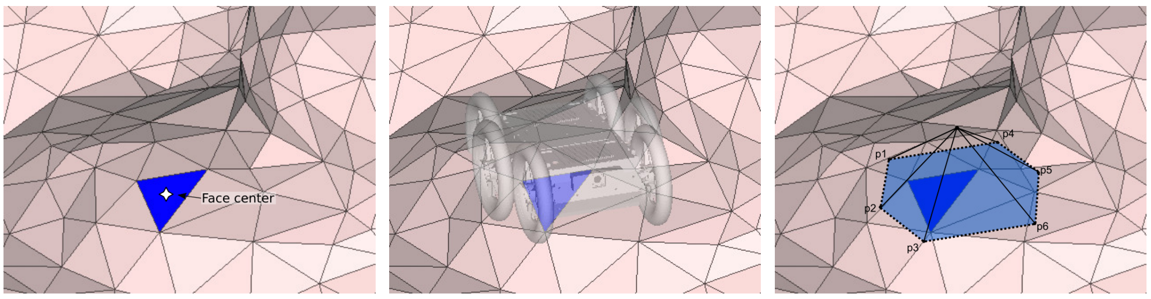
3.2.4.2 Terrain Traversability Metric

The terrain traversability metric helps to find the flattest path from start to goal positions. This metric defines the cost function $C_2(p)$ based on the positive angle $T(n)$ between the mesh's face normal vector ($\vec{\mathbf{n}}^z$) and the canonical Z-axis ($\vec{Z}$), such as:

$$T(n) = \arccos\left(\frac{|\vec{\mathbf{n}}^z \cdot \vec{Z}|}{||\vec{\mathbf{n}}^z||\,||\vec{Z}||}\right), \quad C_2(p) = \sum_{n \in \bar{p}} T(n). \tag{3.2}$$

In contrast to the other metrics, here we propose the use of multiple terrain interaction proposals to estimate the $\vec{\mathbf{n}}^z$ of the robot estimated position: (i) the robot represented as a point, (ii) a complete 3D model interaction with the terrain and (iii) a support polygon interaction. Figure 3.11 shows the multiple interaction models.

When the robot is represented as a point lying in the center of a mesh's face, we can directly use the Eq. 3.2, since no extra calculations are needed (Figure 3.12).



(a) Single point. (b) 3D model. (c) Support polygon.

Figure 3.11: Different robot-terrain interactions: (a) face center as the single point interaction, (b) complete robot 3D model interaction, and (c) support polygon interaction.

For the complete 3D model and support polygon representation, other methods must estimate the robot's final pose with the terrain. An in depth description of those other terrain interaction models, and the proposed strategy for selecting the proper model can be seen in Appendix B.

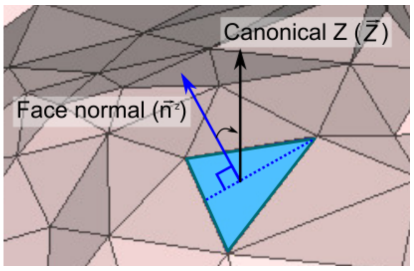


Figure 3.12: Example of the face normal angle of a node $\vec{\mathbf{n}}^z$ related to the canonical $\vec{Z}$.

3.2.4.3 Energy Consumption Metric

This metric aims to compute the path that leads to the minimum energy consumption from the start to the goal. For simplification purposes, we considered that the robot moves with constant velocity; this means that the robot spends energy while accelerating and braking to keep the uniform movement.

We use the energy $E(n)$ estimation method proposed by Santos et al. [2019], where the required energy to move the robot from neighboring nodes $n-1$ and n is estimated as a linear regression of the battery consumption and the terrain inclination, the friction coefficient, robot mass, angle θ between the vector linking the mesh centers, resulting in the cost function $C_3(p)$. E_{avg} is the mean battery consumption while turning 2π rad and a, b are the linear regression parameters. The quantities $\phi(n)$ and $dist(\text{n})$ are estimates of the angular and linear displacements, respectively, when the robot moves between nodes $n-1$ and n. Finally, we define $C_3(n)$ as:

$$E(n) = \left(\frac{E_{avg}\ \phi(n)}{2\pi} \right) + (a\ \theta(n) + b)\ \text{dist}(n), \quad C_3(p) \ = \ \sum_{n \in \bar{p}} E(n). \tag{3.3}$$

where the constants are defined as:

$$\begin{aligned} E_{avg} &= 37735.9 \\ a(\theta) &= \begin{cases} -475.1, & \text{if } \theta \leq 0 \\ 564.97 & \text{otherwise,} \end{cases} \\ b(\theta) &= \begin{cases} 1089.3, & \text{if } \theta \leq 0 \\ 1364.9, & \text{otherwise.} \end{cases} \end{aligned} \tag{3.4}$$

The value for the constants were defined by the linear regressions for downhill and uphill consumption rates presented in Santos et al. [2019].

3.2.4.4 Combining All Metrics

Considering the conflicting objectives, we propose a cost function based on all the multiple metrics previously mentioned during the path planning, where the robot operator can set a trade-off between them through weights. Therefore, the algorithm evaluates each metric's relevance while finding paths connecting the start and goal positions. Thus, the cost function is given as follows:

$$C_4(p) = \sum_{n \in \bar{p}} \left[P_d N_d D(n) + P_t N_t T(n) + P_e N_e E(n) \right], \tag{3.5}$$

where P_d, P_t and P_e are the weights that set the metric priorities related to distance traveled, terrain traversability, and robot energy consumption, respectively. The scalars N_d, N_t e N_e are normalization coefficients to guarantee the metric values lies between $[0, 1]$. The normalization coefficients are calculated by estimating the maximum and minimum values for all the metrics in the traversability graph, using the following equation:

$$\text{normalization}(v, v_{max}, v_{min}) = \begin{cases} 0, & \text{if } v_{max} \leq 0 \text{ and } v_{min} \leq 0 \\ \dfrac{v - v_{min}}{v_{max} - v_{min}}, & \text{otherwise.} \end{cases} \tag{3.6}$$

When exploration time is a priority, the operator can set P_d with a higher value than the others. In environments with a high risk of tipping over, P_t requires a higher priority. On the other hand, in situations where energy consumption is critical, P_e should be greater. The sum of these weights has unit value, $\sum_{i\in\{p,t,e\}} P_i = 1.0$.

The weighted sum, or also called *aggregating functions*, was proposed and validated by Coello [2000] as a naive but efficient method for multi-objective optimization. Other methods such as the weighted min-max or Tchebysheff norm could also be used in a similar way [Miettinen, 2012; Chang, 2015; Jeddisaravi et al., 2016].

3.2.5 Path Navigation

The path planner works as a finite state machine (Figure 3.13), where the robot needs to reach the final path waypoint (or another terminal state) to plan a new path. The mesh reconstruction algorithm must get the most extended point cloud possible – generally from the final location of the previous path – to allow the mesh to cover the robot's maximum reachable range and maximize the next waypoint's distance from the current robot position. Reaching the waypoint before generating a new path prevents generating short paths for a limited view of the environment and estimates fewer and longer paths. The terminal states initially proposed are related to collisions, robot locomotion problems, or if the goal is reached.

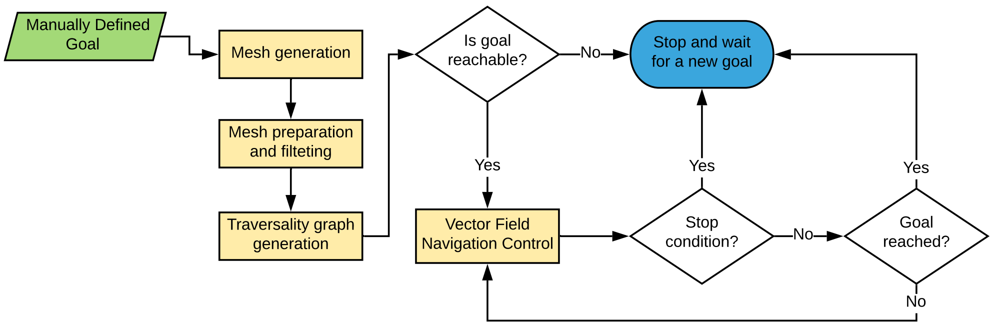

Manually Defined Goal
Mesh generation
Mesh preparation and filteting
Traversality graph generation
Is goal reachable?
No
Stop and wait for a new goal
Yes
Vector Field Navigation Control
Stop condition?
Yes
No
Goal reached?
Yes
No

Figure 3.13: Navigation state machine.

An artificial vector fields-based controller performs the low-level navigation [Gonçalves et al., 2010]. The control methodology consists of a function $\mathbf{F}(\mathbf{p}) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that defines a reference velocity F for the robot at each point for the path p. This velocity is responsible for guiding the robot towards a reference path by computing linear v and angular ω velocities via a Feedback Linearization, where a reference velocity is attributed to a virtual point located at a distance d of 0.2 m, in the forward direction from the robot's rotational center to overcome the non-holonomic restrictions of the EspeleoRobô platform. This control method was validated for indoor locomotion in Rezende et al. [2020].

3.3 Experiments and Results

The proposed mapping and navigation pipeline was evaluated through simulated and real-world experiments with the EspeleoRobô in representative scenarios of rugged outdoor environments, subterranean caves, and other indoor spaces. The three-dimensional goal location on the experiments was determined manually using RViz.

3.3.1 Virtual environments

All experiments executed in virtual environments were performed using the simulated version of the EspeleoRobô [Azpurua et al., 2019] inside the CoppeliaSim[‡] simulator (v4.0.0), executed with ROS Kinetic and Ubuntu 16.04. The experiments are executed within the virtual framework for robots in confined spaces presented in Cid et al. [2020].

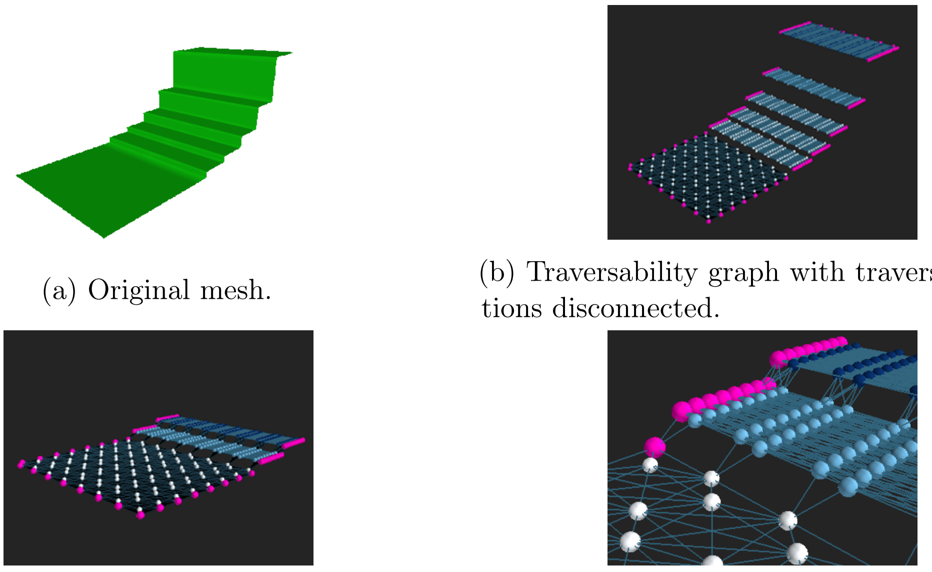
3.3.1.1 Traversability graph generation

An example of the traversability estimation with a threshold of $\theta_{max} = 30°$ over a synthetic multi-level scenario is depicted in Fig. 3.14. The stair-like structure gives a good reference to reachable and non-reachable multi-level platforms. Hence a connected graph is generated for every traversable platform, but as the connection between platforms has a greater inclination than the established threshold, the platforms are not initially connected (Figs. 3.14a-b). After the bumpiness post-processing step, the first two platforms within the traversable range are connected to the main base (Figs. 3.14c-d).

3.3.1.2 Path planning in rugged terrains

We used a representative simulated environment to compute paths connecting multiple points through obstacles and uneven terrains to validate the path planner. Figure 3.15a shows the scene considered for the simulation. The terrain corresponds to a Motocross field in Ouro Preto - MG, Brazil (20°24'05.4"S, 43°30'58.0"W), whose model was obtained with photogrammetry reconstruction techniques with RGB images acquired by a drone and scaled to the correct scale with GPS data. Figure 3.15b shows the point cloud used to compute the cost functions of each metric described. The red,

[‡]`http://www.coppeliarobotics.com`



(a) Original mesh.

(b) Traversability graph with traversable sections disconnected.

(c) Reachable traversable platforms connected.

(d) Zoom-in to the connections between platforms.

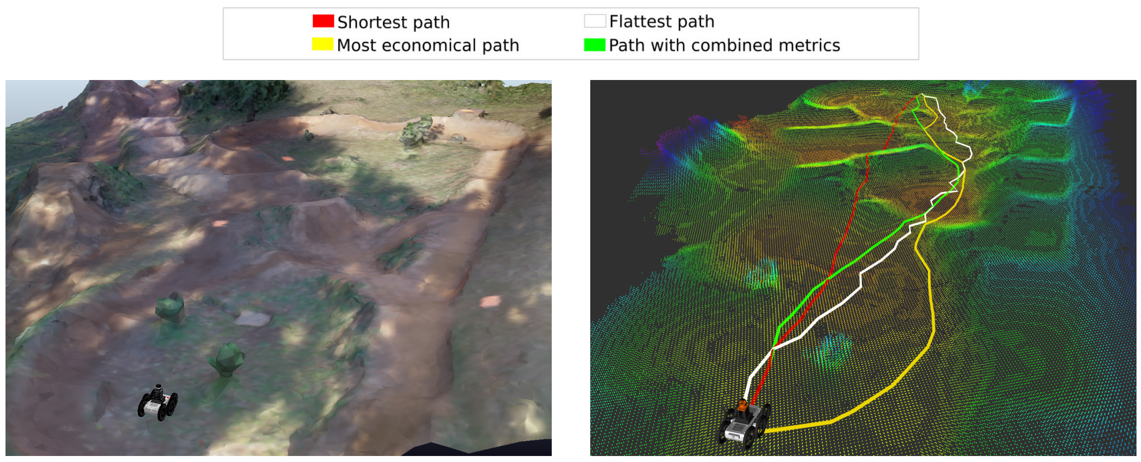
Figure 3.14: Ground traversability map generation: (a) synthetic mesh reconstruction with slopes at different heights, (b) traversability graph, (c) traversable map with bumpiness threshold applied, and (d) zoom of the connected edges.

white, and yellow paths correspond to the shortest, flattest, and most economical paths, respectively. The green path was obtained with the combined cost function. The paths behave as expected, as the shortest path tends to perform as a direct line to the goal, bypassing obstacles and other risk areas of the map. On the other hand, the most economical and flat paths tend to minimize terrain roughness and high slopes. The gains of the combined metric were defined as: $P_d = 0.5$, $P_t = 0.25$ and $P_e = 0.25$.

3.3.1.3 Terrain interaction models

We performed a performance analysis of the multiple terrain interaction models we proposed in the previous sections: Terrain angle estimation directly from a mesh face normal, the pose optimization algorithm using the robot's stability polygon, and a full physics simulation using Pybullet. The analysis also includes the terrain selection method proposed in Algorithm B.3. The environment used for this experiment is a subset of the DARPA Cave scenario as shown in Figure 3.16, where the different terrain interaction models were used to generate the path minimizing terrain roughness (flattest path from goal to destination).

The paths shown in Figure 3.16 have a similar visual footprint. This similarity is confirmed by the analysis of point o point angle differences between the three techniques on all map faces, as shown in Figure 3.17. In this sense, the normal angle estimation gives



(a) Simulated environment in CoppeliaSim. (b) Point cloud with estimated paths.

Figure 3.15: Path planning validation in simulated reconstruction of a rugged outdoor environment.

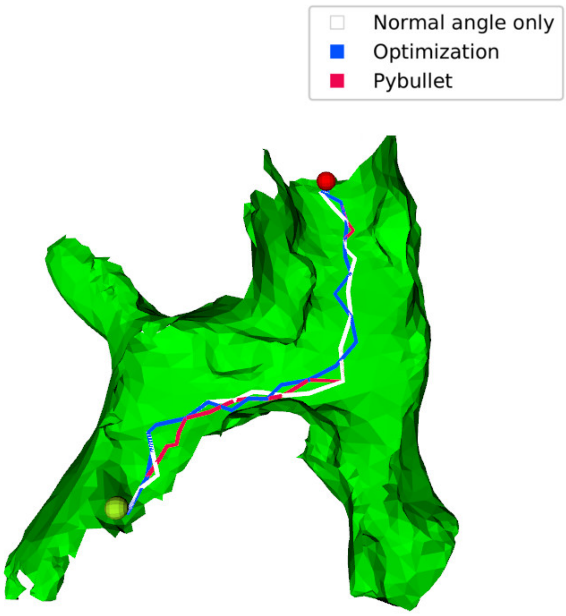


Figure 3.16: Validation of different terrain interactions models over a section of the DARPA Cave environment (path length ≈ 22 m). The white path is generated with only the normal face information, while the blue path is estimated with an optimization algorithm and the red one is estimated by a realistic physics simulation with the terrain and the robot's model.

more approximate results to the Pybullet method than the Optimization estimation (5.11 vs. 6.99 degrees).

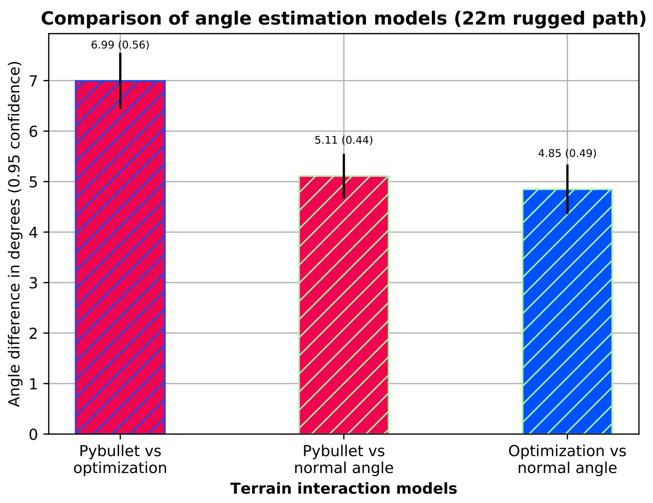


Figure 3.17: Path planning validation in simulated reconstruction of a rugged outdoor environment.

The more accurate method for estimating the robot's pose over the rugged terrain is using the Pybullet physics simulator. With Pybullet, the interaction of the terrain topography with the robot can be performed in a realistic environment considering the complete three-dimensional model of the robot, gravity, collisions, and friction coefficients. However, accurate simulations come with an increased computational cost. Since the pose estimation is needed for every mesh face that the path search algorithm uses for expansion, this could lead to a potential processing bottleneck, especially considering the limited computing resources available in the robot. In this sense, the time spent on each model, including the estimated time reduction of the model selection algorithm, is shown in Figure 3.18. The first three bars are the normal angle extraction (very fast at 0.06 seconds), followed by the optimization method with 33.86 seconds, and the slowest being the Pybullet method taking 107.84 seconds for the same path of $\approx$ 22 meters. The last two bars show a considerable time decrease between the Optimization and the Pybullet-based angle estimation when using the selection method, prioritizing the faster normal extraction over the other more accurate methods where all the neighbors of a face share the same relative angle.

A visual comparison of the Optimization and Pybullet generated paths when using the model selection method can be observed in Figure 3.19. The paths using the selection method are closely similar to those without it but with higher computational costs. The more lightweight methods are not too different estimating the robot's pose

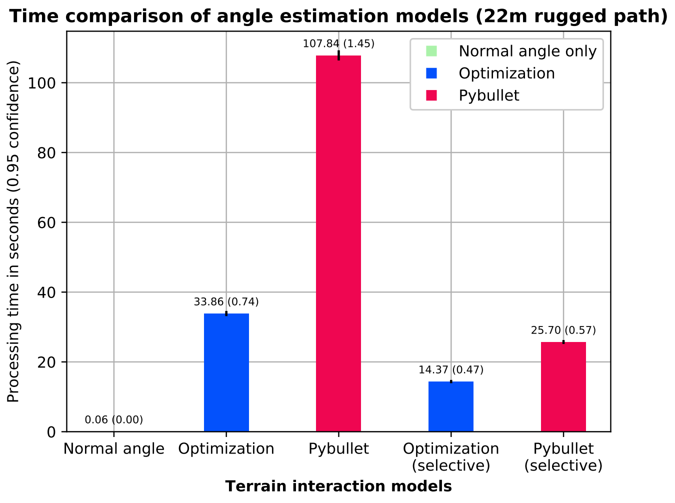


Figure 3.18: Estimated pose estimation time for each of the terrain interaction models, including the selection algorithm.

than the comprehensive physics emulation with Pybullet. When terrain accuracy is a vital requirement and CPU-intensive methods such as Pybullet are used, the model selection algorithm could decrease computational cost while maintaining a similar final pose estimation.

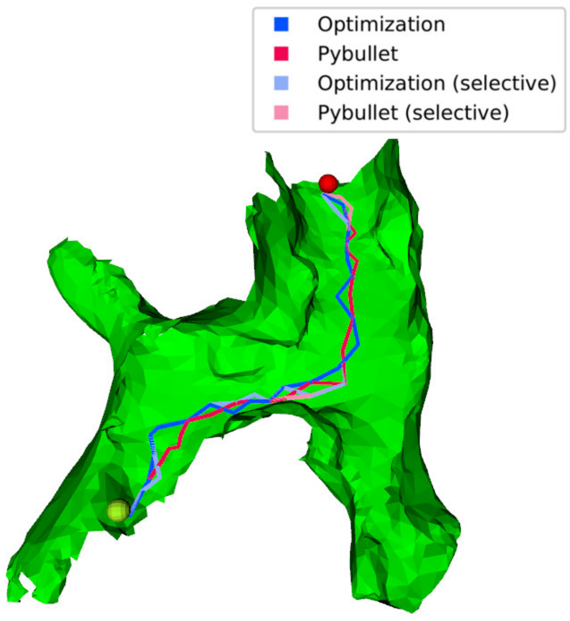
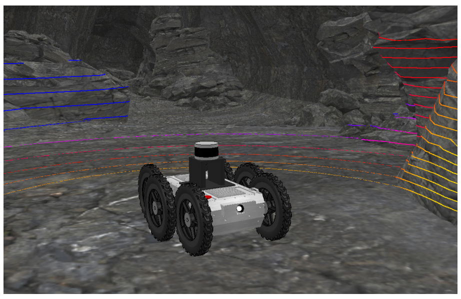
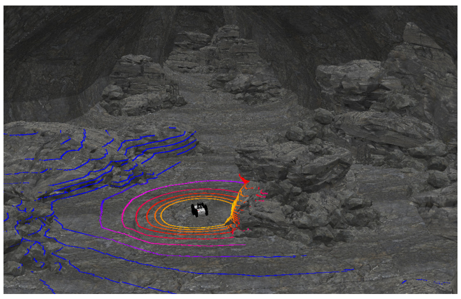


Figure 3.19: Validation of the terrain interactions models with the model selection algorithm over a section of the DARPA Cave environment (path length ≈ 22 m). The dark and light blue path is estimated with an optimization algorithm and the selection method, and the dark and light red ones are estimated by a realistic physics simulation with the terrain and the robot's model and the selection method.

3.3.1.4 Simulated DARPA Cave Scenario

To validate the navigation pipeline in a safe underground scenario, we used the subterranean cave environment from the DARPA Subterranean Challenge[§]. This environment has obstacles, uneven terrain, and realistic subterranean cave geometry (Figure 3.20).


(a) Simulated EspeleoRobô with LiDAR.


(b) Inside view of the cave environment.

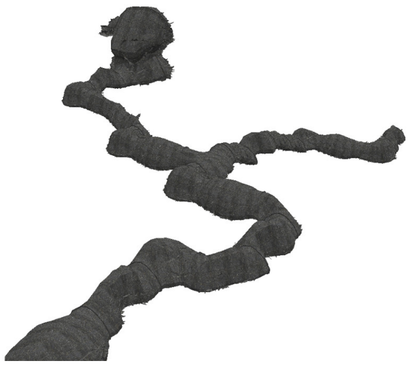
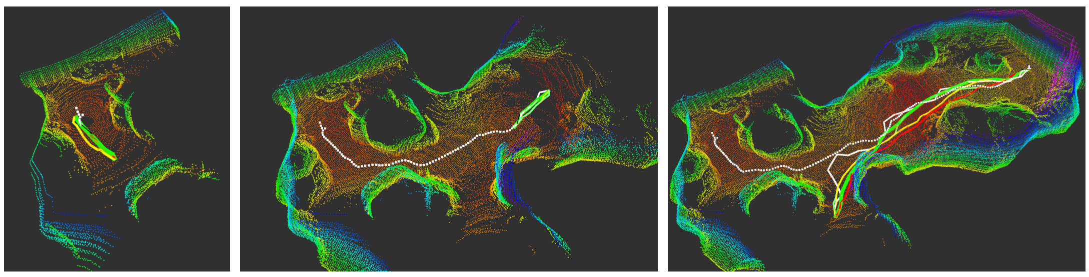

(c) Segment of the map where the experiments were performed.

Figure 3.20: Virtual experimental cave setup: (a) Simulated EspeleoRobô, (b) inside view of the cave, and (c) external view of the cave map.

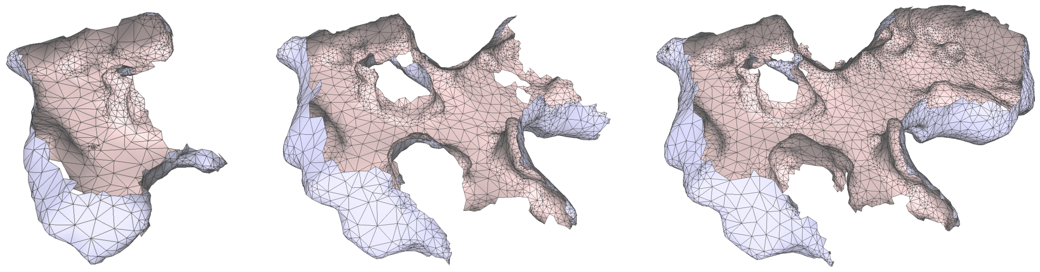
The complete pipeline of navigation, mapping, localization, and control was evaluated on a simulated exploration mission for over $\approx 100m$, as shown in Figure 3.21. In this experiment, the robot performed path planning using the LiDAR mapping algorithm's point cloud. This point cloud is converted to a 3D mesh, and the paths are calculated using this mesh. The robot's localization method is given only by the LiDAR SLAM algorithm; no global positioning system was used.

The sequential evolution of the map and the 3D meshes can be observed in Figures 3.21a and 3.21b. Finally, in Figure 3.21c the final reconstructed environment is depicted with the odometry data estimated by the LiDAR-SLAM algorithm (white dotted line), and an example of the different path planning metrics in solid colors. During the planning step of the exploration, the robot used the combined metrics strategy

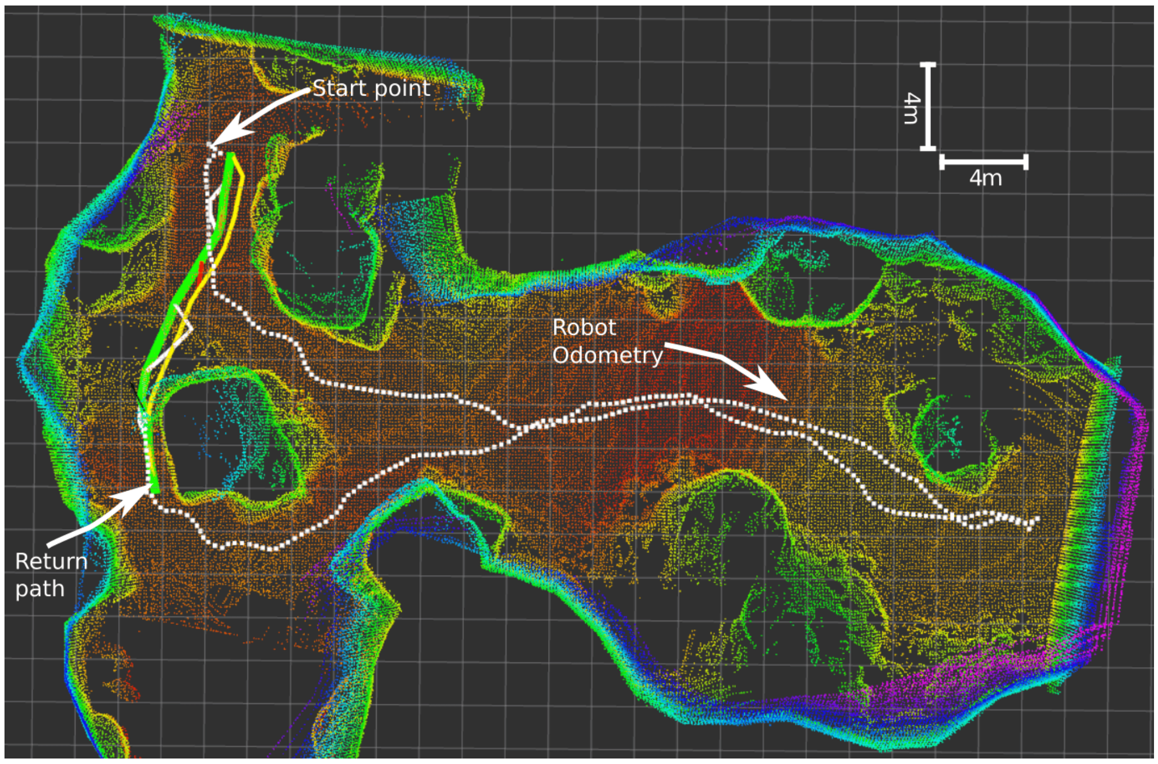
[§]"cave_02" from `https://github.com/osrf/subt`



(a) Sequential process of mapping and navigation.



(b) Resulting map meshes for path planning at the previous steps.



(c) Top view of the complete map, depicting one instance of path planing (colored lines). Odometry is signalized as white dots.

Figure 3.21: Inspection pipeline experiment in a simulated DARPA cave environment ($\approx 100m$). Top: sequential coverage and the corresponding point clouds (a) and map meshes (b) used to calculate the paths. Bottom: top view (c) of the complete map. The robot odometry is depicted in a white dotted line. The routes for every step are denoted in solid red (distance), yellow (energy consumption), white (traversability), and green (combined metrics) lines.

defined in equation (3.5) to navigate (solid green line) with $P_d = 0.25$, $P_t = 0.50$ and $P_e = 0.25$. The density of the LiDAR-SLAM point cloud was defined as 20cm.

3.3.2 Real World Results

Real experiments were also performed with the EspeleoRobô platform in two representative scenarios: (i) a subterranean gold mine and (ii) an indoor multi-level scenario. In both experiments, we use a high-range directional 900MHz antenna for communication. The robot was equipped with an Ouster OS1 LiDAR (16 lines), Xsens MTI-G-710 IMU, and an Intel RealSense Depth D435i camera.

The robot followed the combined metrics path (green path in Figures 3.23c and 3.24c) and the weights were defined with $P_d = 0.80$, $P_t = 0.10$ and $P_e = 0.10$. The density of the point-cloud for the LiDAR-SLAM algorithm was defined at 5 cm.

3.3.2.1 Autonomous Navigation in a Subterranean Mine

The subterranean mine experiment was performed at the Mina du Veloso gold mine, located in Ouro Preto – MG, Brazil (20º22'34"S, 43º30'57"W). In Figure 3.22 the environment and robot setup can be observed. Mina du Veloso is a 400-year-old colonial gold mine with almost 300 meters of narrow multi-level corridors, with rugged terrain and strong magnetic interference.

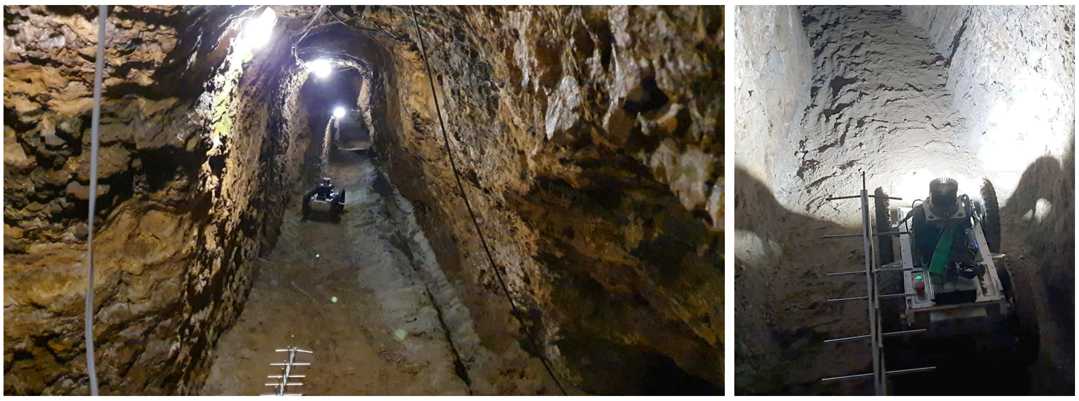
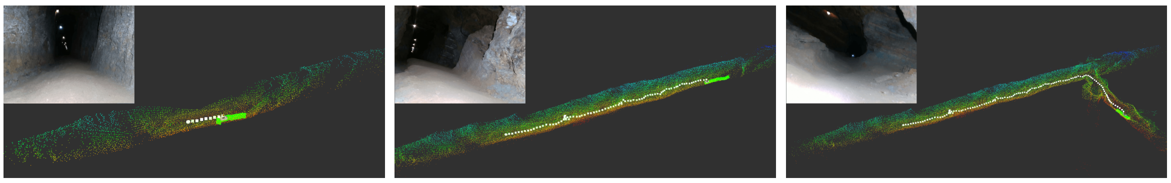

(a) Wide angle of the cave environment. (b) Robot setup.

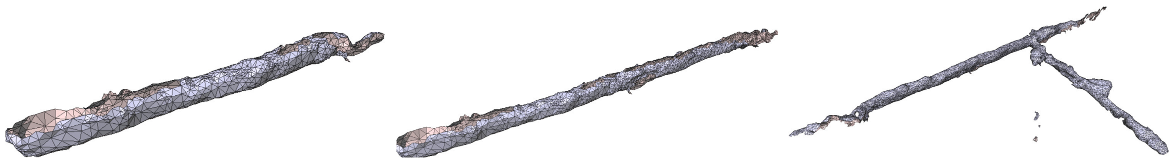
Figure 3.22: Experimental setup of the EspeleoRobô at the Mina du Veloso gold mine.

The navigation and inspection experiment results inside the Mina du Veloso gold mine can be observed in Figure 3.23, where the robot performed mapping over ≈55 m of connected cave tunnels autonomously.

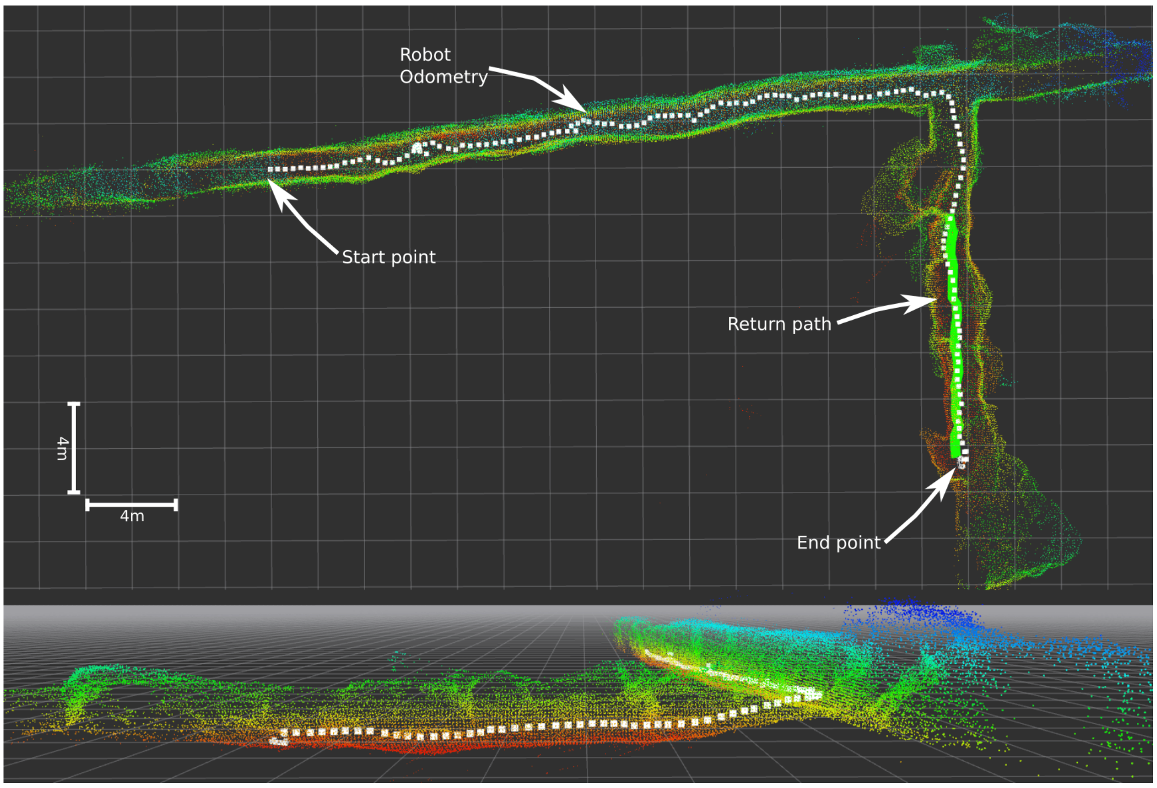
Figure 3.23a depicts the sequential point clouds generated by the LiDAR-SLAM algorithm, including a picture of the current scenario taken from the frontal RGB camera



(a) Sequential process of mapping and navigation, including an RGB image of the robot environment.



(b) Resulting map meshes for path planning at the previous steps.



(c) Top and side view of the complete map, depicting one instance of path planing (green line). Odometry is signalized as white dots.

Figure 3.23: Inspection pipeline real experiment at the Mina du Veloso gold mine ($\approx 55m$). Top: sequential coverage (a) and the correspondings map meshes, (b) used to calculate the paths, with corresponding RGB images of the environment. Bottom (c): final point cloud of the exploration pipeline (top and lateral view). In (c) the LiDAR odometry of the robot for the entire inspection mission is depicted in a white dotted line, and the navigation path for the combined metric is shown in green.

of the robot. Figure 3.23b shows the sequential mesh generation used for navigation and path planning. An overview of the complete map, including odometry, can be observed in Figure 3.23c. Only the combined metric path (solid green line) was estimated in this experiment, given the previous validations' results. This scenario was particularly challenging as the terrain was rugged and slippery, with narrow corridors including multiple small bumps, holes, and rocks. In this sense, the planning algorithm's weights were defined to prioritize straighter collision-free paths to prevent any extra in-place rotations. The increased cloud resolution (5 cm) used in this experiment helped detect and reconstruct the complex terrain more reliably, at the cost of an increased point cloud size.

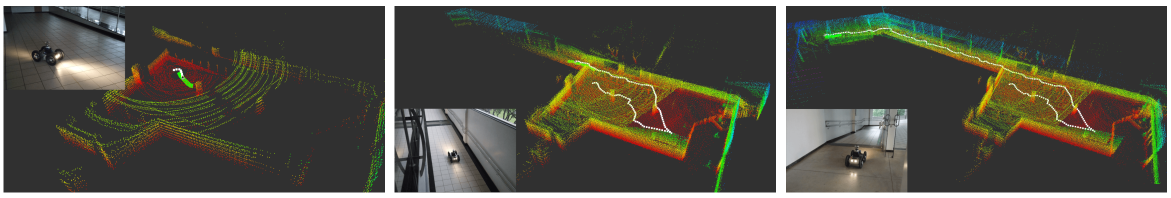
3.3.2.2 Autonomous Navigation in an Indoor multi-level Scenario

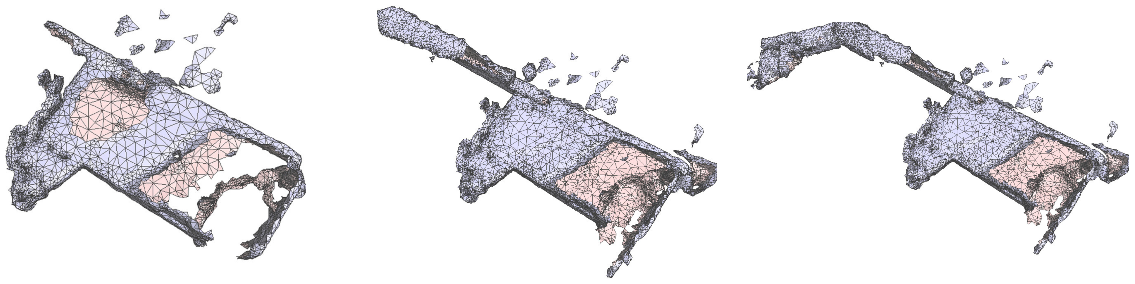
The experiment of the multi-level indoor scenario was performed in the Engineering School building at the UFMG campus in Belo Horizonte - MG, Brazil (19°52'09.2"S, 43°57'45.3"W). The inspection and navigation results for this experiment can be observed in Figure 3.24. In this scenario, the robot explored two separate floors joined by an inclined and narrow corridor. The robot performed over $\approx$80 m of autonomous navigation. This scenario was challenging due to the small and inclined corridor the robot must take to reach another floor, which validates the capacities of the 3D navigation of the proposed pipeline for narrow passages.

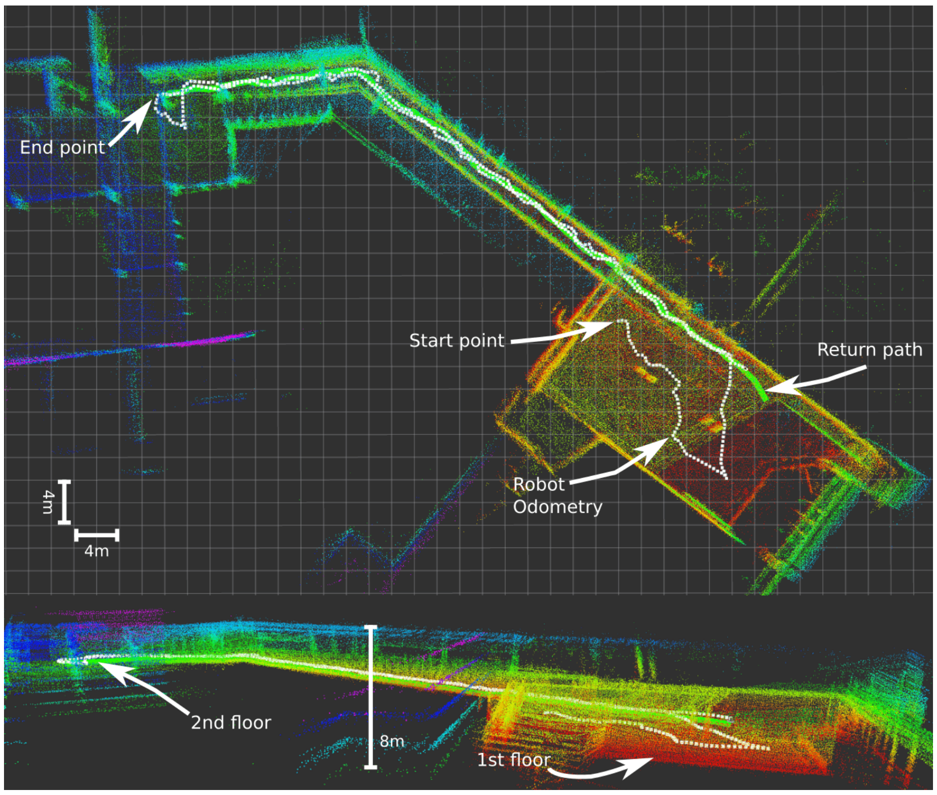
This experiment was the hardest for the proposed pipeline. Despite this indoor scenario presented mostly flat surfaces connected with an inclined path, the surfaces were shiny, with the presence of multiple windows and glass doors, which generated noise in the LiDAR sensor readings. Considering the extended range of LiDAR sensors (more than 150 m) and that the triangulation methods used for mesh generation are susceptible to outliers, the use of the raw point cloud that could be reflected on surfaces and extends over windows generates unwanted noise and delays on the mesh generation algorithm. In this regard, we implemented noise reduction techniques, such as artificially limiting the LiDAR sensor range and performing a grid clustering using KD-trees. After setting up those filters over the raw LiDAR data, the meshing algorithm correctly reconstructed the navigable areas, and traversable paths were successfully generated.

3.3.3 Planning performance analysis

To evaluate the navigation pipeline's performance, we measured the mesh reconstruction time for multiple point clouds extracted iteratively from the LiDAR-SLAM algorithm when performing an exploration mission using a laptop with an Intel i7-4810MQ CPU


(a) Sequential process of mapping and navigation, including an RGB image of the robot environment.


(b) Resulting map meshes for path planning at the previous steps.



(c) Top and lateral view of the final map, depicting one instance of path planing for returning to the home base (green line). Odometry is signalized as white dots.

Figure 3.24: Mapping, navigation, and control pipeline real experiment at an indoor multilevel scenario ($\approx 80m$). Top: sequential coverage (a) and the corresponding map meshes (b) used to calculate the paths, with corresponding RGB images of the environment. Bottom (c): final point cloud of the exploration pipeline (top and lateral view). In (c) the LiDAR odometry of the robot for the entire inspection mission is depicted in a white dotted line, and the navigation path for the combined metrics is shown in green.

with 16GB of RAM. Figure 3.25 shows the mesh generation times for an indoor scenario that extends to over ≈80 meters. Every data point is the mean of 3 reconstructions, and the standard deviation is depicted as a light blue shadow. It is possible to observe that the first meshes are generated relatively fast in less than 20 seconds. The largest cloud takes 96 seconds, allowing the algorithm's execution directly on the robot's embedded computer online. The results show a near-linear relationship between the cloud size and the time to perform the reconstruction.

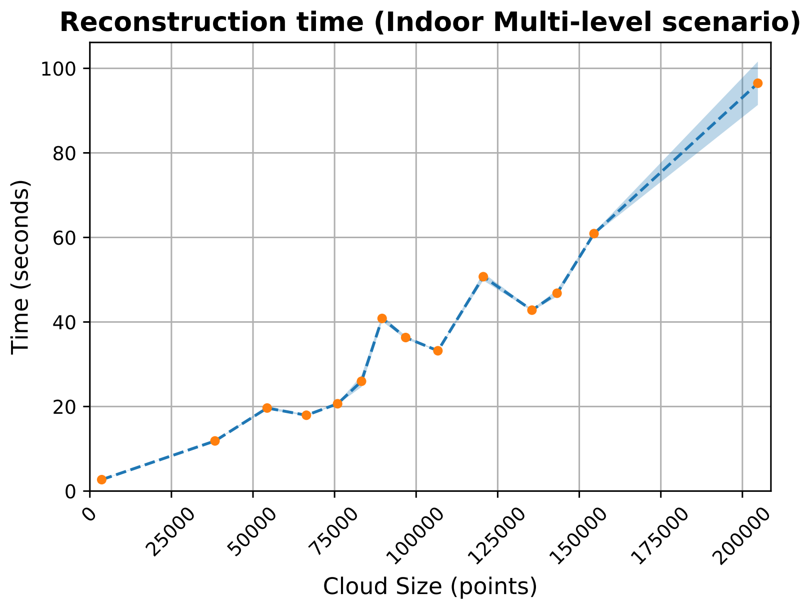


Figure 3.25: Mean reconstruction time for the mesh algorithm over multiple iteratively generated point clouds for a real exploration experiment on an indoor scenario that extends for over ≈80 meters. Every data point is the mean time of 3 runs. The standard deviation is depicted by the shadow in light blue.

The planning step, performed after the mesh reconstruction from the point cloud, transforms the mesh on a graph where the Dijkstra search algorithm finds the more suitable path from the current robot's position to the desired endpoint considering multiple terrain metrics. Figure 3.26 shows the time spent searching the path over an increasing graph size at variable path lengths. Results showed that the time spent in planning is more related to the graph's size than the path size, as shown by the linear time growth despite the variable path size (color of the dots). This procedure was relatively fast, at a maximum of ≈18 seconds for the worst-case on a complete exploration experiment. The time spent on path planning could be increased or decreased, given the graph density, original point cloud size, or mesh filtering.

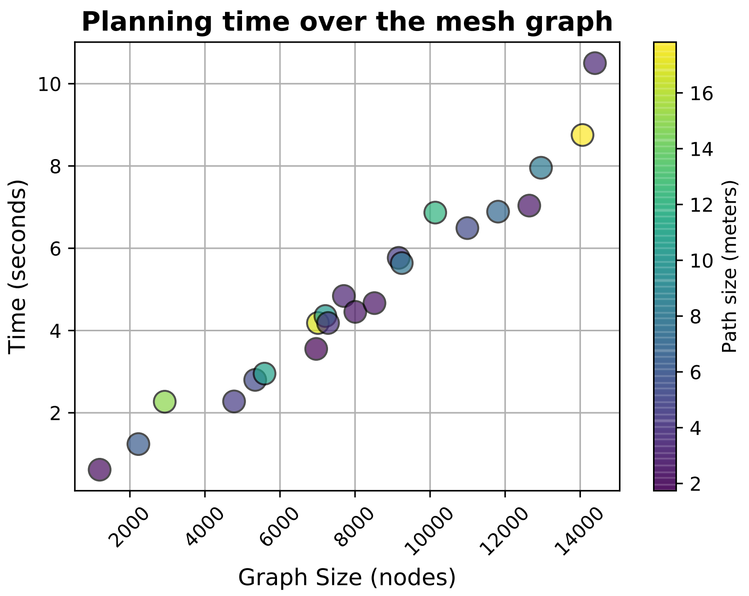


Figure 3.26: Mean planning time for the navigation algorithm over multiple iteratively generated environment graphs. Every data point is the mean time of 3 runs. The color bar represents the path size.

Chapter 4

Terrain-aware autonomous exploration

"Somewhere, something incredible is waiting to be known."

— Carl Sagan.

Autonomous robots must solve multiple complex challenges to explore the real world. The characteristics of confined and subterranean spaces lead to the need for detailed three-dimensional maps and efficient and safe path planning and navigation, particularly for ground robots. Despite current trends with aerial robotics, ground robots have the advantage of an increased payload and endurance over aerial platforms and are generally more mechanically robust, making them still the primary choice for many industrial situations. By leveraging on the path planning method showed in Chapter 3, here we propose a method for the automatic selection of the next-best view, considering the expected volumetric gain of the frontiers and the cost of traversing to them; we answer the question "Where is the next best goal?" (Figure 4.1).

Figure 4.1: Chapter question: "Where is the next best goal?" (for one robot).

4.1 Problem Formulation

We tackle the problem of an autonomous exploration with no initially prior information in confined environments, such as subterranean mines, tunnels, and caves. The task will be executed by a single ground robot R, where its pose q is represented by a configuration $\mathbf{q}_{\mathrm{k}} \in \mathrm{SE}(3)$.

The robot must map a static environment $\mathcal{E} \in \mathbb{R}^3$, which poses critical challenges for the navigation, for example, obstacles, uneven terrain, and narrow passages. Let $\mathcal{M}$ be a three-dimensional occupancy grid representation of $\mathcal{E}$, generated by the observations of a 3D range sensor. The map will be initially set to $\mathcal{M} = \mathcal{E}_{unknown}$, as we do not assume any previous information on the environment. Space already explored ($\mathcal{E}_{known}$) can be either mapped into $\mathcal{M}_{free}$ (visited cells that do not contain any obstacle at the time of measurement) or $\mathcal{M}_{occupied}$ (cells with more than 0.5 probability of occupation given the sensor model). Given an initial position $n_i \in \mathcal{M}_{free}$, to reach a goal position n_{goal} we must define a safe and efficient continuous path $p = \{n_1,\ n_2,\ ...,\ n_n\} \rightarrow \mathcal{M}_{free}$, such that $p_0 = n_i$ and $p_n = n_{goal}$.

Finally, a fundamental aspect of the exploration task is the selection of a location to visit. Therefore, based on the concept of frontier, given a collection of reachable frontiers $\mathcal{F}$, we must select the one that looks more promising to aggregate information to our map.

Problem 2 (Three-dimensional Terrain Aware Exploration). *Given a ground robot R in a confined static environment $\mathcal{E}$. The problem consists of efficiently build a map $\mathcal{M}$ (3D occupancy grid) of the environment. For that, we must:*

- *Create and maintain updated a set $\mathcal{F}$ of all current identified reachable frontier regions in $\mathcal{E}$;*
- *Select a frontier $f_i \in \mathcal{F}$ that maximizes the information gain and is relatively close to the robot's current position;*
- *Determine a feasible and collision-free path p that drives the robot to an open area near the selected frontier.*

4.2 Proposed method

The proposed methodology initially considers a 3D 360° range sensor, such as a multi-line LiDAR, and an IMU to perform the SLAM used to generate a 3D occupational grid using the LeGO-LOAM algorithm [Shan and Englot, 2018; da Cruz Júnior et al., 2020]. Other sensors with a more limited view could also be modeled given the intrinsic sensor parameters. As shown in the previous chapter, the selected SLAM algorithm was validated in GPS-denied scenarios and can run online using the robot's embedded computational resources. Even though LiDAR data is sparse, the long-range and illumination invariance make it one of the best sensors for exploration in confined and subterranean spaces, particularly for path planning since it is possible to acquire a significant amount of terrain data in a few timesteps. However, the exploration algorithm must be clever enough to ignore the spaces and uncovered spots left by the sensor that are most probably uninteresting areas for visitation. In this sense, an exploration algorithm that does not deal with this problem correctly will spend precious time covering local frontiers instead of exploring other more interesting areas of the environment, thus taking longer to generate an adequate global map.

As proposed in the terrain-aware navigation (Chapter 3), a mesh reconstructed from the estimated occupational grid serves as input for generating a traversability graph with the robot's reachable regions. Nevertheless, in this case, we are also using Octrees to represent the environment's occupied and free areas. This representation of the environment is handy to perform ray casting and other topological analyses. In this regard, the information gain and the path cost are calculated for every extracted frontier using both an octree and a reconstructed mesh. Finally, the selected frontier is navigated by the robot following an estimated safe path, and this cycle repeats until there are no more frontiers to explore or any other stop condition is met. A high-level description of the proposed method for exploration is depicted in Figure 4.2. When there are no more frontiers to visit, the robot should return to the location where the mission started (base station).

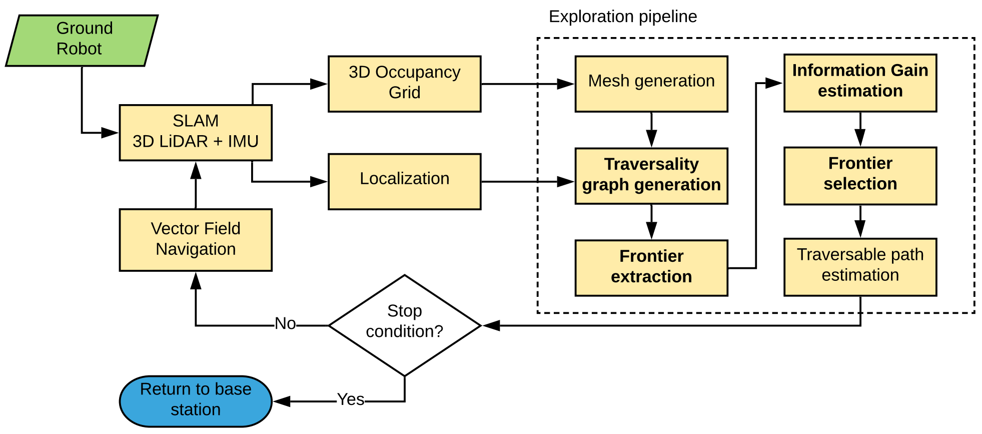


Figure 4.2: High-level description of the proposed multi-step procedure for autonomous exploration.

4.2.1 Reachable frontiers extraction

The traversability graph $\mathcal{G}$ and the complete mesh $\mathbb{M}$ are used to estimate the map frontiers. The known traversable areas that are neighbors of unexplored regions, including regions that are edging the map's borders without being an obstacle, are considered frontiers. Therefore a face in the mesh is considered as a frontier if:

$$\text{isFrontier}(\mathbb{M}, i) = \begin{cases} 1, & \text{if } |N_{\mathbb{F}}(i)| \leq 2 \\ 0, & \text{otherwise,} \end{cases} \tag{4.1}$$

where $N_{\mathbb{F}}(i)$ is the set of neighboring faces of face i.

The extracted frontiers that do not also belong to the traversability graph are removed. This way, unreachable frontiers are eliminated from the ground robot's exploration pipeline but can be marked as *unreachable regions* for future cooperation with robots with other locomotion dynamics –such as aerial platforms. The frontier points are clustered into groups by their Euclidean distance using the Density-based spatial clustering algorithm (DBSCAN) [Schubert et al., 2017], considering a distance (eps) and minimum group size ($\mathcal{C}_{min}$). Finally, we use the KD-tree search to determine the visit location of a frontier cluster as the closest reachable point in $\mathcal{G}$ to the cluster's centroid. The frontier estimation process is described in Algorithm 2, where $\mathcal{B}$ are the raw mesh frontiers, $\mathcal{C}$ the clusters of reachable frontiers, and n a reachable node from $\mathcal{G}$ used as frontier visit point.

Algorithm 2: Reachable frontiers extraction

```
$\mathcal{B} \leftarrow \emptyset$                                    ▷ Raw mesh frontiers
for $i \leftarrow 1$ to $|\mathbb{F}|$ do
    if $isFrontier(\mathcal{M}, i)$ then
        $\mathcal{B} \leftarrow \mathcal{B} \cup \mathbb{F}_i$
    end
end
$\mathcal{B}' \leftarrow traversabilityFilter(V, \mathcal{B})$
$\mathcal{F} \leftarrow \emptyset$                                    ▷ Reachable frontiers centroids
$\mathcal{C} \leftarrow DBSCAN(\mathcal{B}', eps, \mathcal{C}_{min})$
for $c \in \mathcal{C}$ do
    $n \leftarrow getCentroidReachableNode(\mathcal{G}, c)$
    $\mathcal{F} \leftarrow \mathcal{F} \cup n$
end
return $\mathcal{F}$
```

4.2.2 Information-theoretic frontier selection

To estimate the utility of a frontier, we use the mutual information gain metric over the 3D map $\mathcal{M}$, and the cost of the path $C(p)$ to reach the frontier. The mutual information uses the probability of the octree cells to calculate the current entropy [Shannon, 1948] of the map and then compares it with the expected entropy of the map after performing a virtual exploration at the selected frontier. The binary entropy is defined as:

$$H(\mathcal{M}) = -\sum_{i,j,k} p_{ijk} \ \log(p_{ijk}), \tag{4.2}$$

where $\mathcal{M}$ is the current map, and p_{ijk} is the outcome of the Bernoulli random variable representing the cell occupation.

The mutual information $I(\mathcal{M}, f_i)$ is used to obtain the expected information gain by visiting frontier f_i, i.e.:

$$I(\mathcal{M}, f_i) = H(\mathcal{M}) - H(\mathcal{M}|f_i), \tag{4.3}$$

where $H(\mathcal{M}|f_i)$ is the expected new entropy.

As the robot *does not know* what is behind the frontiers, the posterior probability of occupation could be difficult without actually visiting the environment. A solution for this problem is to perform a *virtual exploration* of the frontier given the information we have gathered so far: (i) the location, disposition, and layout of a three-dimensional frontier, and (ii) intrinsic parameters of the LiDAR sensor the robot is using for SLAM, such as the field of view and range.

The *virtual exploration* phase uses the sensor model maximum range and field of view to project rays in the current map $\mathcal{M}$ considering the vehicle will be in the centroid of the face representing each frontier visitation point in $\mathcal{F}$. Since we assume a sensor with a 360° horizontal FoV, the robot's orientation is not used for calculations, only the mesh face's inclination at the frontier visiting point is used for transformation purposes (expected robot's roll and pitch).

Given the current map $\mathcal{M}$ state, the projected rays give a reasonable estimate of the maximum free volume the robot could sense of a frontier. Rays that hit walls or fall within the range of known empty cells ($\mathcal{M}_{known}$) do not increment much information. We consider rays that escape known areas and approach the maximum sensor range without hitting an obstacle as more informative. Since the measurements from these types of multi-line LiDAR sensors are sparse, we performed a filling step at the virtual map using a uniform sampling of mesh surfaces $\mathbb{F}$, preventing the projected rays from escaping through walls and solid objects over the occupancy grid $\mathcal{M}$.

Finally, the metric for selecting the best frontier is:

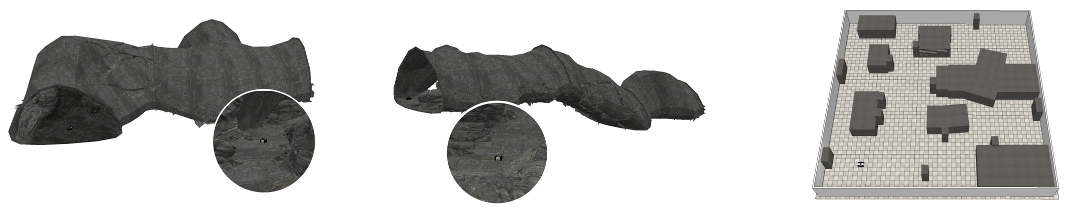
$$c^* = \underset{\forall f \in \mathcal{F}}{\arg\max} \frac{MI(\mathcal{M}, f) + e}{C(p^f)}, \tag{4.4}$$

where p^f is a path from the robot's current position to frontier f, and e is a small tolerance constant.

4.3 Experiments and results

All experiments were executed in virtual environments using the simulated version of the EspeleoRobô [Azpurua et al., 2019] inside the CoppeliaSim* simulator (v4.0.0), executed with ROS Kinetic and Ubuntu 16.04. The experiments are also executed within the virtual framework for robots in confined spaces presented in Cid et al. [2020]. The laptop used for the experiments has an Intel i7-4810MQ CPU and 16GB of memory.

We validated the performance of the proposed exploration pipeline using three scenarios: (a) a single-level cave map extracted from the DARPA SubT challenge, (b) a multi-level cave map also from the DARPA challenge, and (c) a synthetic simplified cave map in a flat environment. The scenarios can be observed in Figure 4.3. The subterranean environments present realistic challenges for locomotion, such as uneven terrains, multi-level platforms, rocks, and other obstacles. The EspeleoRobô robot was equipped with a simulated Ouster OS1 LiDAR sensor and an IMU. For all experiments, the maximum slope is defined as $\theta_{max} = 30°$ and the bumpiness as 25 cm.



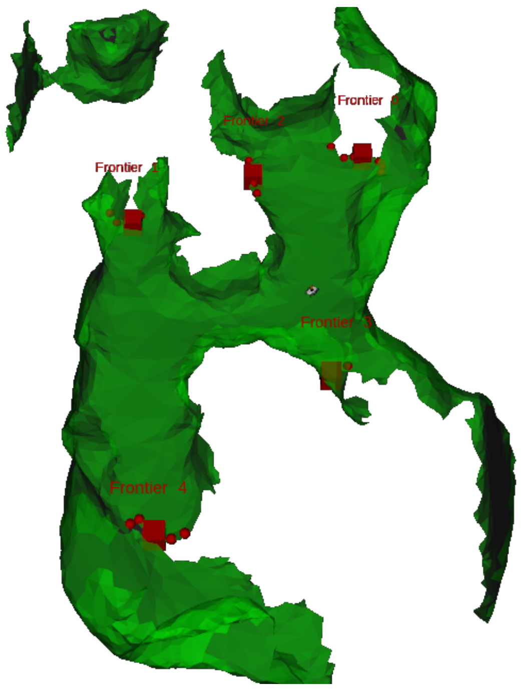
(a) Single level cave scenario. (b) Multi-level cave scenario. (c) Synthetic cave map.

Figure 4.3: Simulated environments used for experimental evaluation: (a) a single-level cave map extracted from the DARPA SubT challenge, (b) a multi level cave map also from the DARPA challenge, and (c) a synthetic simplified cave map in a flat environment [Howard and Roy, 2003].

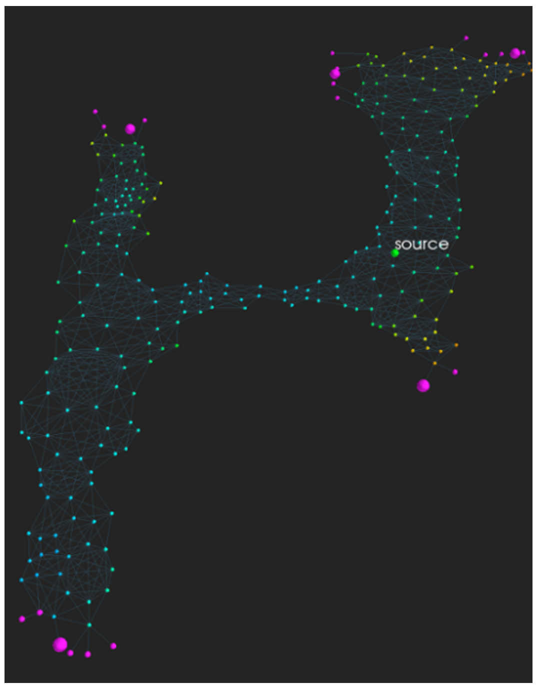
*http://www.coppeliarobotics.com

4.3.1 Frontier extraction

An example of the frontier extraction over a simulated DARPA SubT cave section using the mesh and the traversability graph can be observed in Fig. 4.4. In Figure 4.4a, the red squares mark the expected locations of ground frontiers, and in Fig. 4.4b, the traversability graph that leads to the frontiers visit points in pink. This double verification reduces false positives and guarantees that the estimated frontiers can be reachable by the ground robot.



(a) Reconstructed cave mesh with estimated frontiers (red).

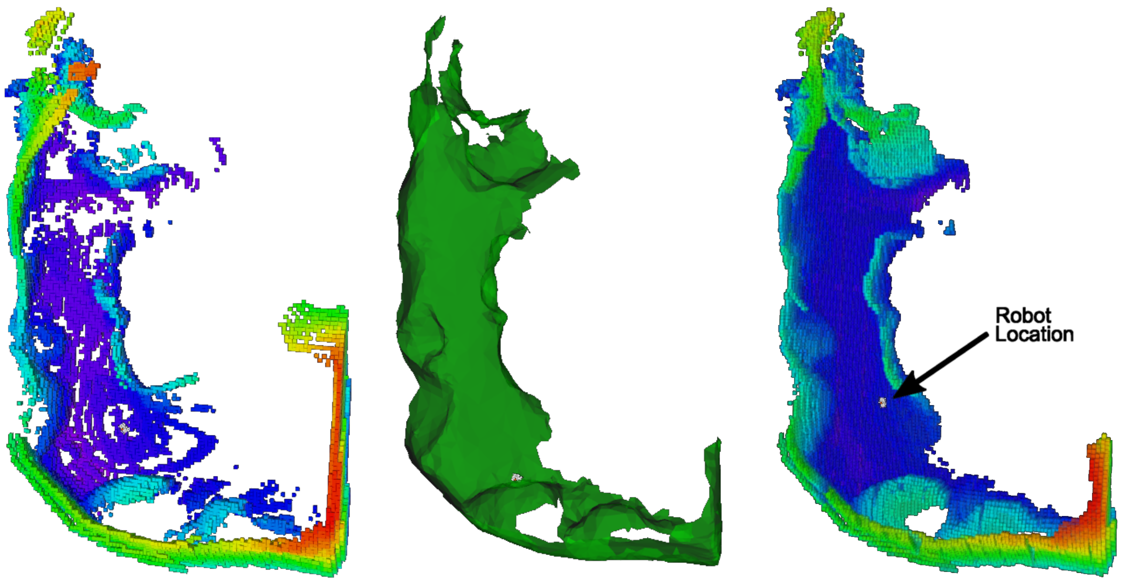


(b) Traversability graph with the remaining traversable frontiers (pink).

Figure 4.4: Frontier estimation for the reconstructed mesh and traversability graph. (a) Generated mesh with extracted frontiers (red), and (b) traversability graph plot with the remaining frontiers from the mesh that are also within the graph. The resulting frontier candidates are clustered using the DBSCAN algorithm and, therefore, valid for exploration (pink).

4.3.2 Octree filling

The raw map generated by the LeGO-LOAM mapping algorithm [Shan and Englot, 2018] is a sparse octree that is not directly suitable for many operations evolving terrain analysis and information estimation. Given the high number of spots in $\mathcal{M}$ with missing $\mathcal{M}_{occupied}$ cells, as shown by Fig. 4.5a, the ray-tracing algorithm will not work as best because the rays could pass through the missing cells and give a wrong volumetric measurement. The mesh reconstruction procedure will fill those small missing parts of the map and generate continuous surfaces that we sub-sample and use to fill the octree, as shown in Figures 4.5b-c.



(a) Sparse octree. (b) Reconstructed mesh. (c) Filled virtual octree.

Figure 4.5: Filling missing parts of the octree with mesh information, preventing the projected rays from escaping for holes in walls and solid objects: (a) raw estimated octree from the LiDAR SLAM algorithm, and (b) reconstructed mesh, and (c) the filled virtual octree.

4.3.3 Complete autonomous exploration

The exploratory efficiency over the cave scenarios (Figure 4.3) was validated by comparing the resulting mapped point clouds generated iteratively by the SLAM algorithm with a reference map. Small discrepancies in the point clouds could occur since only the embedded robot sensors were used for localization. Therefore, the comparison method *sim* is defined as:

$$sim(a, b) = RMSE(a, b) * 0.5 + RMSE(b, a) * 0.5, \tag{4.5}$$

where a and b are the generated pointcloud and reference cloud, and the Root Mean Square Error (RMSE) between the clouds is given by:

$$RMSE(a,b) = \sqrt{\frac{\sum_{i=0}^{|b|} dist(a_i, b_i)^2}{|a|}}. \tag{4.6}$$

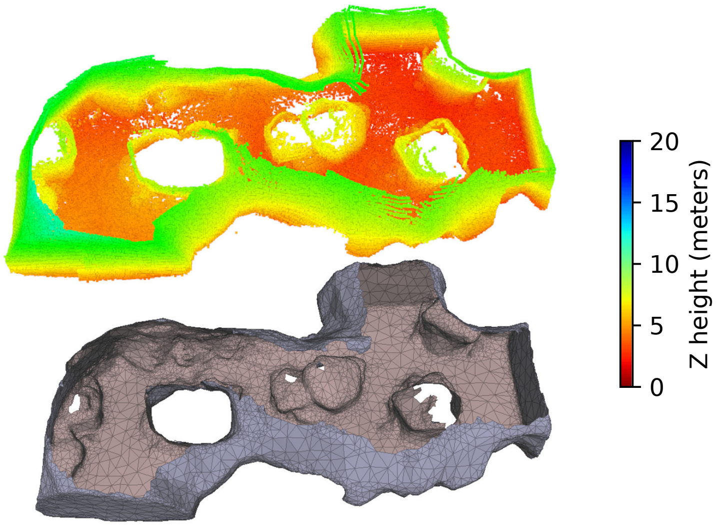
The resulting maps and the reconstructed meshes for every scenario could be observed in Figures 4.6 (online video available[†]).

We validated multiple frontier and path selection metrics: (i) our proposed approach using the information gain and traversability path generation of the frontiers to select the next visit point, (ii) a greedy selection of the closest next frontier using the path that gives the smallest Euclidean distance instead of the terrain aware one, and (iii) random selection of the frontier using a terrain aware path. All methods use the traversable graph to generate the paths, but in the case of the greedy approach, only the shortest distances are prioritized regardless of terrain roughness or energy consumption. The results of this analysis can be observed in Figure. 4.7, where the displayed lines are the mean RMSE for every timestep of ten repetitions per experiment. The timesteps were estimated using only the time the robot moved and did not include the paths' computing time. The proposed metric (blue) converges to a lower error rate at every environment than the other metrics, even if the more terrain-friendly paths generated by the proposed metric are longer than the shortest euclidean path. An interesting effect of the greedy metric is that the shortest path usually increments the chances of entering a riskier area, also increasing the chances of entrapment or collisions. In Figure. 4.7, the timesteps were limited to 1000 and 2000 steps; however, given an infinite amount of time, the curves of all three strategies should converge to an error close to 0.

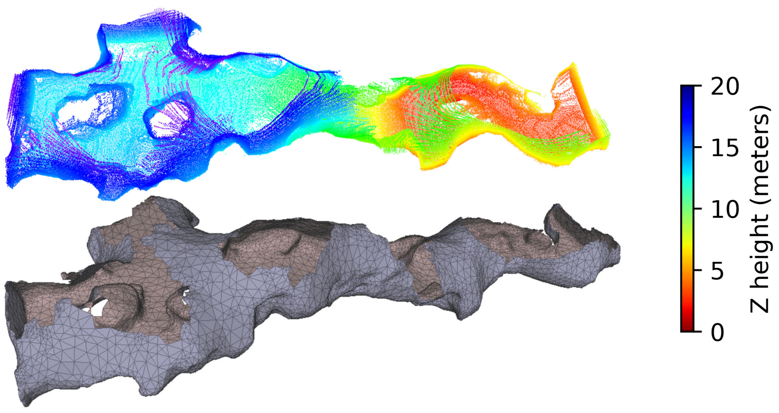
The paths performed by the robot when using the proposed strategy can be observed in Figure 4.8. The robot's exploratory decisions lead to mostly non-repetitive paths where the already covered areas are less likely to be revisited. In this sense, it is possible to observe that the robot consistently takes paths to the more extensive galleries and sections of the map, leaving the minimal rewarding areas for the final stages of exploration, or exceptionally, visit them when they are nearby to the robot's position.

Table 4.1 depicts the maximum processing time taken by the mesh reconstruction algorithm at the final step of the exploration. Showing that the proposed method could be used online with only small pauses of a couple of seconds between planning

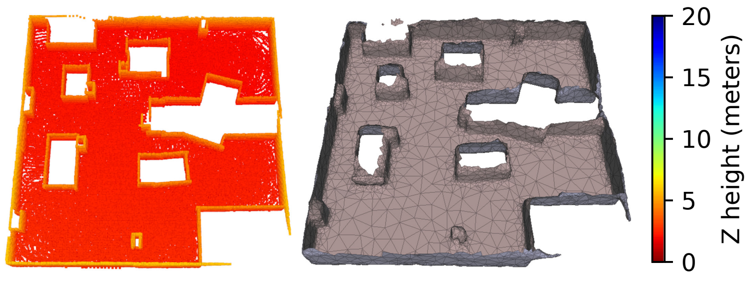
[†]Deterministic Terrain-aware exploration example (video): `https://youtu.be/RrkOvAZEteQ`



(a) Single-level cave.

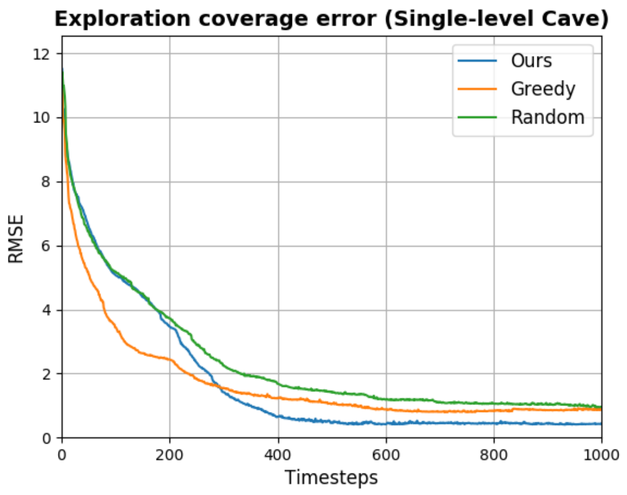


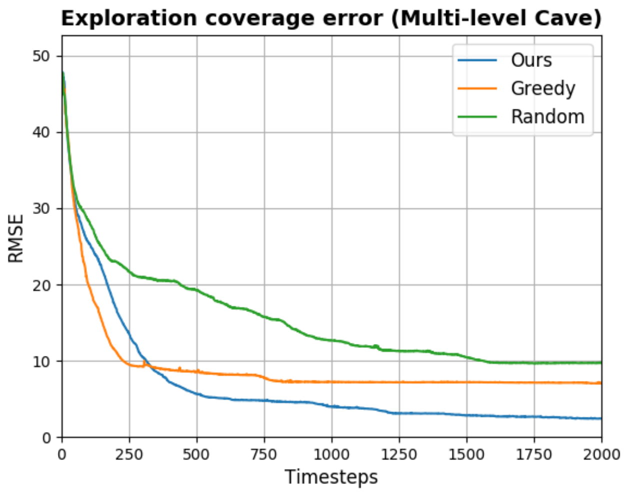
(b) Multi-level cave



(c) Synthetic cave map.

Figure 4.6: Estimated point-cloud (colored) and final reconstructed mesh (brownish) for the cave environments: (a) a single-level cave, (b) a multi-level cave map, and (c) a synthetic cave map. The color gradient of the point-cloud represents the height variation (up to 20m).





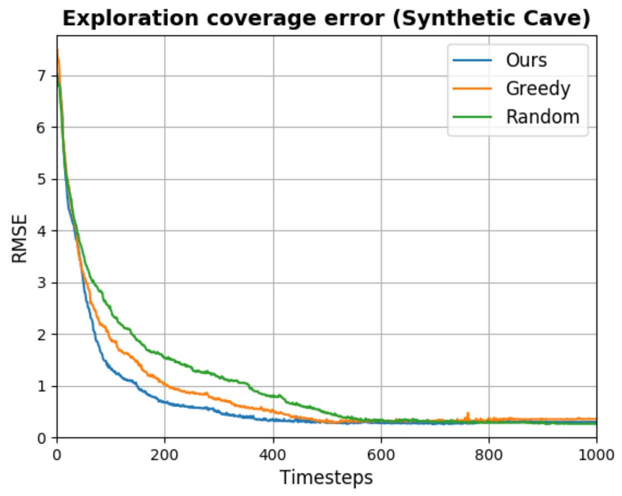
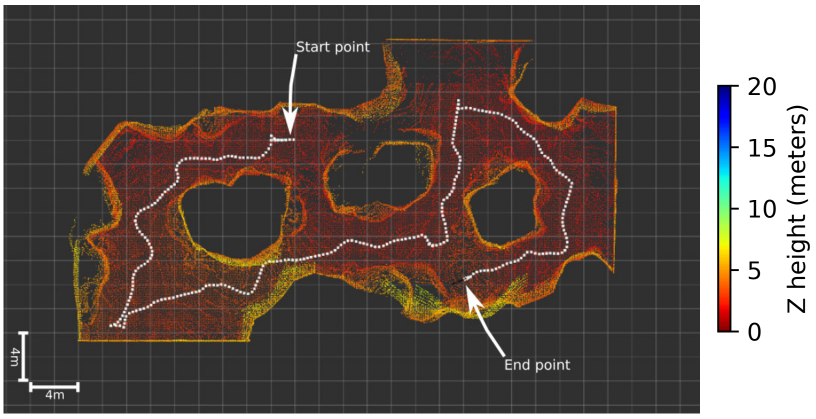
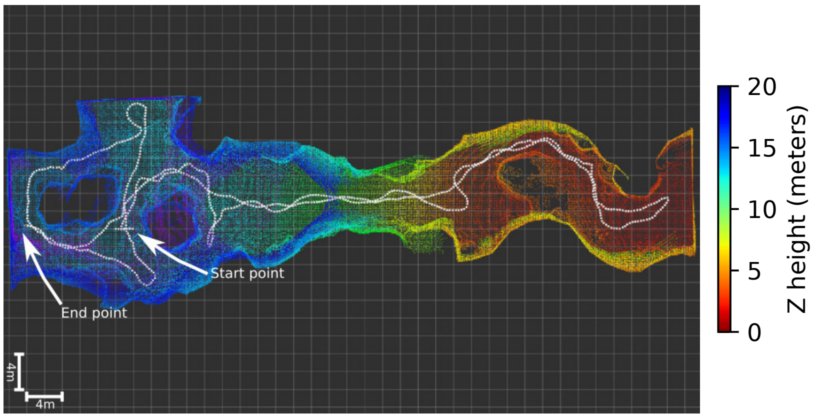


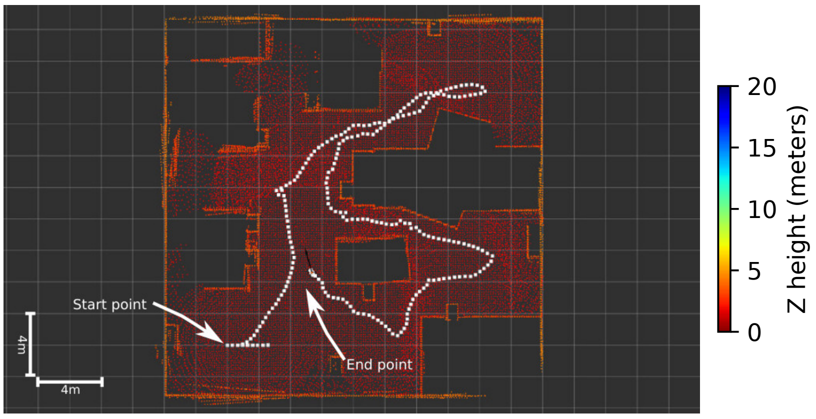
Figure 4.7: Exploration error (RMSE) mean of ten runs comparing the real-time SLAM point cloud with a reference map for the: (a) single-level cave, (b) multi-level cave, and (c) synthetic cave. All methods use the traversability graph to generate the paths. Only the greedy approach prioritizes solely Euclidean distance instead of a smooth, safer path.



(a) Single-level cave.



(b) Multi-level cave



(c) Synthetic cave map.

Figure 4.8: Exploration paths generated by our proposed exploration strategy for the: (a) single-level cave, (b) multi-level cave, and (c) synthetic cave. The white dots shows the robot's LiDAR odometry along the exploration mission.

and execution. The mean time is even lower when the exploration mission is at the beginning since the number of points is also lower.

Table 4.1: Mean execution time (10 runs) of the mesh reconstruction method for all testing environments' final state.

Map	Num. Points	Execution time	σ
Single-level Cave	122084	30.143s	±1.31
Multi-level Cave	142124	38.054s	±1.06
Synthetic Cave	42000	11.648s	±0.72

Chapter 5

Probabilistic exploration

"Accidents happen. That's what everyone says.
But in a quantum universe there are no such things as
accidents, only possibilities and probabilities
folded into existence by perception."
— Dr. Manhattan.

Exact methods for exploration can be computationally complex and, therefore, costly to execute online. Real autonomous exploration robots must address obstacle avoidance to prevent avoidable collisions while selecting an adequate frontier to explore –given the expected information gain and the terrain topography. By improving the exploration method shown in Chapter 4, here we propose a probabilistic approach to path planning and frontier selection that is: (i) faster to compute, (ii) operates with the simpler raw point-cloud data, and (iii) can be used to avoid additive obstacles through a viewpoint filtering step. In this chapter, we answer the question, "How to plan faster?" (Figure 5.1).

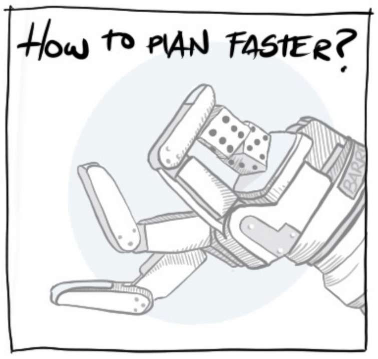


Figure 5.1: Chapter question: "How to plan faster?".

5.1 Problem Formulation

As in Chapter 4, we tackle the problem of an autonomous exploration with no initial prior information in confined environments. Nevertheless, different from Problem 2, which considers mapping a static environment $\mathcal{E} \in \mathbb{R}^3$ with a single ground robot R, here we also consider additive obstacles in $\mathcal{E}$, meaning that the environment is not static anymore. We consider an environment dynamic if there is a chance of the robot encountering an additive environmental modification. An additive obstacle or additive modification of the environment is any addition of points in the map related to a new object that was not initially there. This modification of the environment could happen at any time during the exploration mission. Examples of additive obstacles are collapsed structures, closed doors (previously open), or other elements that could suddenly appear in $\mathcal{E}$.

As before, given a collection of reachable frontiers $\mathcal{F}$, we must select the one that looks more promising, both in navigation easiness and expected information gain.

Problem 3 (Three-dimensional Terrain-aware Exploration with Adaptative Obstacle Avoidance)**.** *Given a ground robot R in a confined dynamic environment $\mathcal{E}$ that could present additive modifications. The problem consists of efficiently build a map $\mathcal{M}$ (3D occupancy grid) of the environment. For that, we must:*

- *Create and maintain updated a set $\mathcal{F}$ of all current identified reachable frontier regions in $\mathcal{E}$;*
- *Determine a feasible global path p_{global} that drives the robot to a frontier;*
- *Quickly determine collision-free local paths p_{local} to guide the robot while performing the global path p_{global}, avoiding dynamic obstacles;*
- *Select a frontier candidate $f_i \in \mathcal{F}$ that maximizes the information gain and is relatively close to the robot's current position.*

5.2 Proposed method

The proposed methodology uses a probabilistic sampling path planner and considers a 3D 360° range sensor, such as a multi-line LiDAR, and an IMU to perform SLAM (with the LeGO-LOAM algorithm [Shan and Englot, 2018; da Cruz Júnior et al., 2020]). Unlike previous approaches, this proposal uses the raw point clouds for planning using a RRT-biased planner (MI-RRT) towards the more informative regions. The use of point clouds instead of the complete reconstruction pipeline for planning improves the efficiency of the terrain analysis at the trade-off of a decreased connectivity precision when generating the traversability graph; however, as seen in the experiments, in most cases, the difference in exploration performance is slight while the computing time is significantly decreased.

A high-level description of the proposed method for exploration with probabilistic path planning is depicted in Figure 5.2. When there are no more frontiers to visit, the robot should return to the location where the mission started (base station).

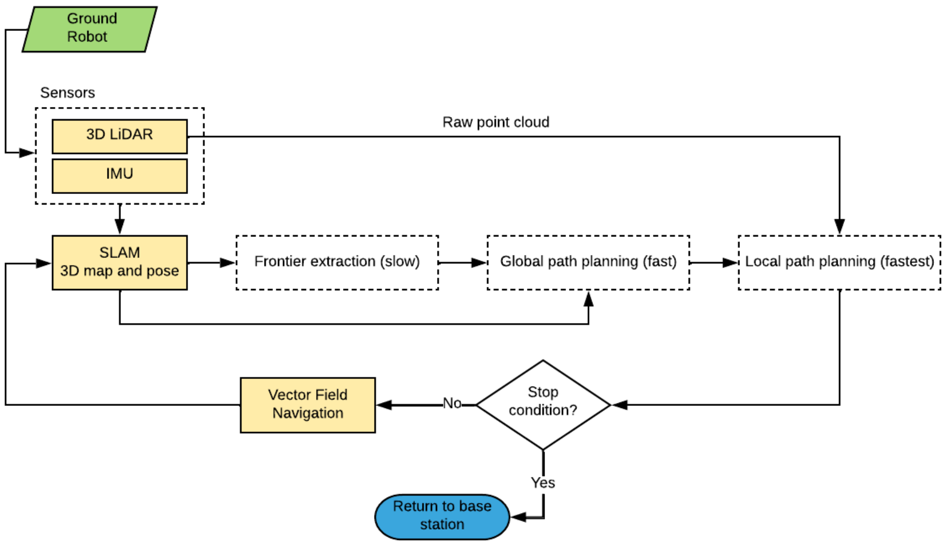


Figure 5.2: High-level description of the proposed method for RRT-exploration capable of local planning to avoid obstacles.

The proposed method is divided into three big steps: (i) frontier extraction, (ii) global path planning, and (iii) local path planning. The frontier extraction procedure (i) uses the same steps as before, extracting a traversability graph from a mesh of the environment, which we use to cluster the reachable frontiers and estimate the information gain of them (Figure 5.3). For more information about this process visit Section 4.2.1.

The global path planning uses a joint point cloud sampled from the mesh and the most recent map point cloud to generate a traversable graph for planning with a biased MI-RRT algorithm. Using the biased version of the RRT algorithm permits the tree generated by the planner to grow into the most informative regions. The traversability graph is generated using a euclidean k-NN algorithm over the filtered raw point cloud. Finally, the local path planning also uses an RRT algorithm, planning smaller paths to traverse the global path without colliding with obstacles. This local planner uses the visible points of the map from the robot's point of view to prevent reaching into occluded areas of the environment.

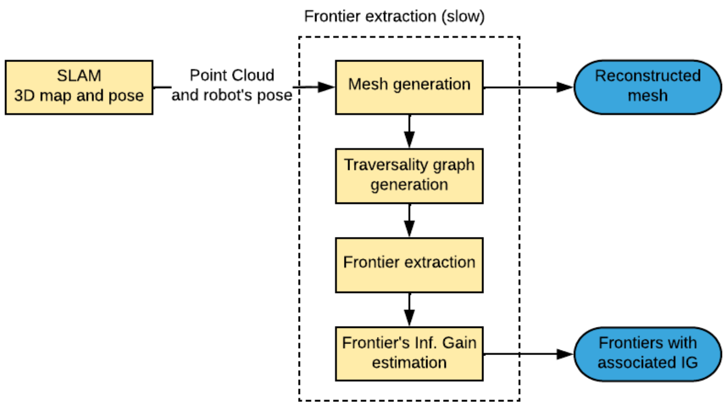

Frontier extraction (slow)
SLAM
3D map and pose
Point Cloud
and robot's pose
Mesh generation
Reconstructed mesh
Traversality graph generation
Frontier extraction
Frontier's Inf. Gain estimation
Frontiers with associated IG

Figure 5.3: Diagram of the frontier extraction process (Expanded from Figure 5.2).

The frontier extraction step is the takes more time than the other ones since we need to reconstruct a global mesh from the point cloud map, and this process can take from seconds to a couple of minutes depending on the point distribution and the presence of anomalies in the map (sensor noise). Given the time it takes, the frontier extraction is performed the least times possible, meaning it is only executed once per exploration cycle. An exploration cycle comprehends the processes of extracting the frontiers, selecting a frontier (via the RRT planner or the exact approach proposed in Chapter 4), and traversing the environment to reach it safely.

On the other hand, the global path planning is faster than the previous step, and it is only performed once after the frontier selection phase; this phase is where the frontier is selected, and the robot estimates a global reference path. The fastest step is the local planning, taking less than a second, and it is executed constantly during an exploration cycle to refine the robot path, avoid dynamic obstacles, and detect out-of-reach situations. Local planning is critical for safe locomotion in a dynamic environment and detecting situations where the robot cannot reach the selected target area –which is not uncommon given that the target frontier is generally selected with only a partial observation of the environment.

5.2.1 MI-RRT global planner with information bias

A high level diagram of the global planning process can be observed in Figure 5.4. The global planner complements the sparse point cloud estimated in the SLAM step with a denser cloud extracted from the mesh generated in the frontier extraction process (Section 4.2.1). Sampling the mesh has several advantages over only using the SLAM point cloud directly, sampling the mesh allows for a denser representation, thus smoothing the generated final paths and filling small holes in the map that have a high chance of being actual traversable terrain. Furthermore, the hole filling and filtering performed in the mesh generation process prevents the robot from spending time covering local frontiers instead of exploring better global targets, thus decreasing the time to generate an adequate global map.

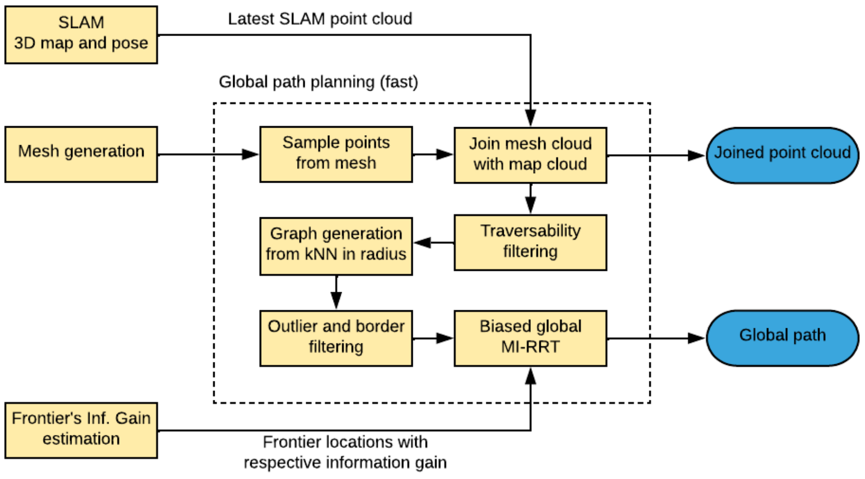
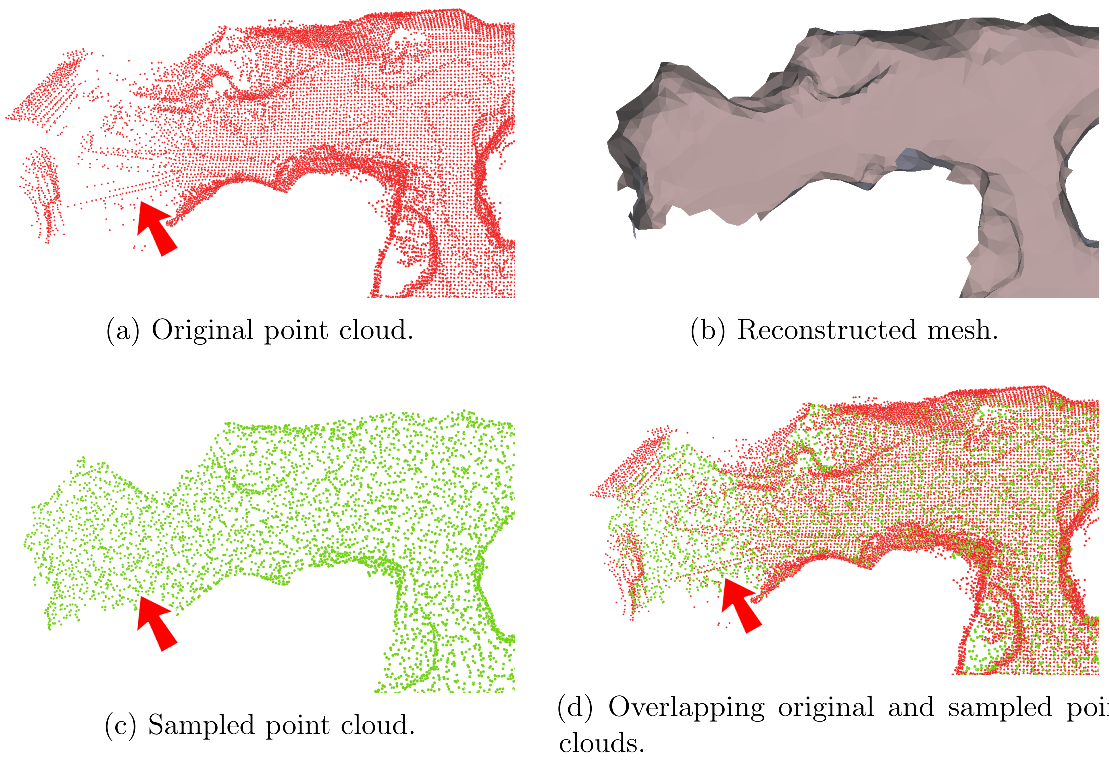

SLAM
3D map and pose
Latest SLAM point cloud
Global path planning (fast)
Mesh generation
Sample points from mesh
Join mesh cloud with map cloud
Joined point cloud
Graph generation from kNN in radius
Traversability filtering
Outlier and border filtering
Biased global MI-RRT
Global path
Frontier's Inf. Gain estimation
Frontier locations with respective information gain

Figure 5.4: Diagram of the global path planning process (Expanded from Figure 5.2).

The sequential process of mesh sampling, showing an under-sampled part of the environment in the original point cloud can be seen in Figure 5.5 (area with the red arrow).

The joint point cloud $\hat{\mathcal{M}}$ is then prepared in several steps to guarantee traversability. The cloud is first voxelized, and then the normals of the points are estimated. Next, the points with a normal greater than the maximum slope angle traversable by the robot (θ_{max}) are filtered. The adjacent points of those defined as not traversable ($\mathbb{X}$) are removed as well by using a search by radius ($\tau_{inflation}$). The points closer to the border are detected using the KD-Tree algorithm [Friedman et al., 1977], which allows performing fast queries in spatial data. After removing the border points, a clustering algorithm (DBSCAN [Schubert et al., 2017]) is used to group the points closer to the initial location of the robot, considering a distance *eps* and minimum group size $\mathcal{C}_{min}$. This clustering allows separate traversable ground points from ceiling points that can be "traversable," but there is no path to reach them.

The final point cloud $\hat{\mathcal{M}}$ then possesses only traversable points, without obstacles and with a safe margin between traversable regions and the obstacles, so the robot does not collide when moving close to the boundaries of the graph. The traversable graph $\mathcal{G}$ is then generated by a k-NN algorithm with a Euclidean distance filter to permit that only closer points can generate edges (filtering by radius of size τ_{bump}). The kNN graph generation process can be observed in Figure 5.6.


(a) Original point cloud.

(b) Reconstructed mesh.

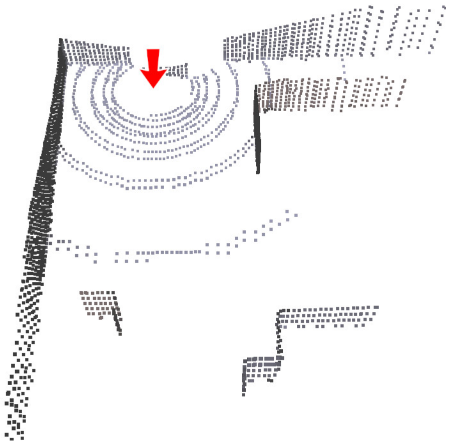
(c) Sampled point cloud.

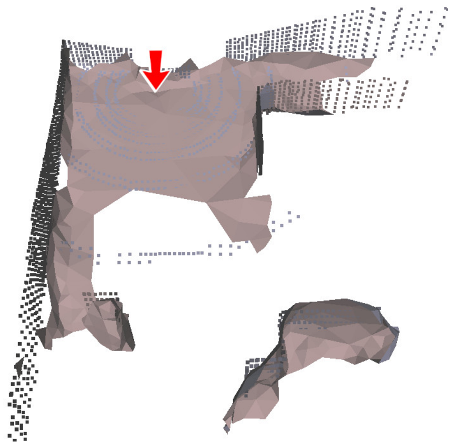
(d) Overlapping original and sampled point clouds.

Figure 5.5: Sequential procedure of sampling points from the reconstructed mesh: (a) initial point cloud in red, (b) reconstructed mesh, (c) sampled point cloud from the mesh in green, and (d) overlapping original and sampled point clouds. The red arrow shows the map area that was filled by the mesh sampling.

The complete process of filtering is depicted in Algorithm 3, where $\mathcal{M}$ is the raw point cloud of the environment generated by the SLAM algorithm.

Previously, we used the exact metric described in Equation 4.4 to select the best frontier among the frontier's set. However, that method required the execution of multiple Dijkstra best path algorithms to generate the best travel cost among frontiers. Here, we propose to adapt the two-dimensional exploration method presented in Pimentel et al. [2018] into a fully 3D environment. This initial exploration algorithm used heuristics to expand the frontiers of a 2D grid map using reasonable priors of structured indoor spaces, and a 2D RRT expanded into the frontiers given the proportional information gain. On the other hand, our work used a fully 3D environment for the information estimation of frontiers considering the terrain roughness of unstructured environments with no priors. As in the previous work, a customized RRT expands to the frontiers given the proportional information gain. This probabilistic method outperforms the previous deterministic frontier selection since it can perform the frontier selection directly at the path planning phase.


(a) Original map point cloud.


(b) Overlapping map point cloud and the reconstructed mesh.

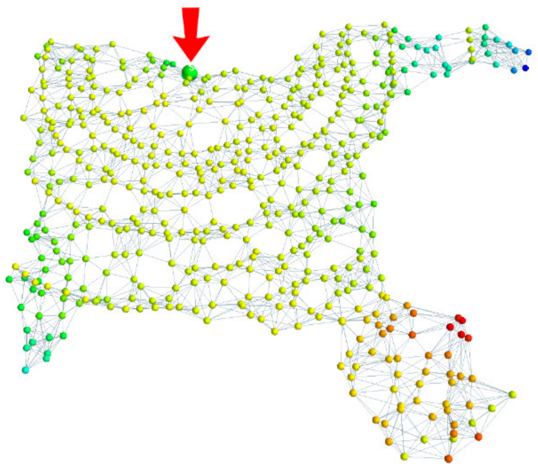

(c) kNN graph ($\mathcal{G}$) representing the traversable regions of the map.

Figure 5.6: Sequential procedure of generating the kNN traversable graph ($\mathcal{G}$). The red arrow shows the location of the robot.

Algorithm 3: Traversability graph generation from point cloud ($\mathcal{M}$)

```
$\mathbb{M} \leftarrow generateMesh(\mathcal{M})$
$\hat{\mathcal{M}} \leftarrow sampleMesh(\mathbb{M})$
$\hat{\mathcal{M}} \leftarrow \hat{\mathcal{M}} \cup \mathcal{M}$        ▷ Join sampled and last measured point cloud
$\hat{\mathcal{M}} \leftarrow voxelize(\hat{\mathcal{M}})$
$\mathbb{X} \leftarrow \emptyset$
for $i \leftarrow 1$ to $|\hat{\mathcal{M}}|$ do
    if $\vec{\mathbf{n}}_i^z < \theta_{max}$ then
        $\mathbb{X} \leftarrow \mathbb{X} \cup \{\hat{\mathcal{M}}_i\}$        ▷ Store obstacle point
        $\hat{\mathcal{M}} \leftarrow \hat{\mathcal{M}} \setminus \{\hat{\mathcal{M}}_i\}$        ▷ Remove point by normal angle
    end
end
$\hat{\mathcal{M}} \leftarrow inflateBorders(\mathcal{G}, \mathbb{X}, \tau_{inflation})$
$\mathcal{C} \leftarrow DBSCAN(\hat{\mathcal{M}}, eps, \mathcal{C}_{min})$
$\hat{\mathcal{M}} \leftarrow filterClusterByOrigin(\mathcal{C}, \{0, 0, 0\})$        ▷ Remove unreachable points
$\mathcal{G} \leftarrow graphFromkNN(\hat{\mathcal{M}}, \tau_{bump})$
return $\mathcal{G}$
```

The RRT algorithm [LaValle et al., 1998], as shown in Chapter 2, is a graph-based sampling algorithm for path planning that is probabilistically complete with an exponential rate of decay for the probability of failure [Frazzoli et al., 2002]. This completeness means that the planner's probability of failing to return a valid path if one genuinely exists decays to zero as the number of samples of the environment approaches infinity.

The proposed RRT-based path search, called Mutual Information RRT (MI-RRT), is described in Algorithm 4. The key of the proposed approach is modifying the sampling part of the RRT planner to use biased point selection based on the information gain of frontiers (*sampleWithRoulette($\mathcal{G}, V, \mathcal{F}'$)* function in Algorithm 4), where $\mathcal{F}'$ is the scaled set of information gain of the reachable frontiers in $\mathcal{G}$ in the $[0, 1]$ range:

$$\mathcal{F}' = \frac{\mathcal{F}}{sum(\mathcal{F})}. \tag{5.1}$$

The roulette sampling uses the normalized information weight of the frontiers to select the next point to add to the tree. We also added an extra fixed probability of a completely random point to decrease possible local minima (τ_{random}). A high-level description of the roulette selection can be observed in Figure 5.7, and the detailed description of the algorithm is shown in Algorithm 5.

The final result of the global planner is a set of ordered vertices to be visited by the robot to reach a promising, informative frontier. This path has a low possibility of

Algorithm 4: MI-RRT path planning algorithm $(\mathcal{G}, \mathcal{N}, \mathcal{F}')$

```
$V \leftarrow \{x_{init}\}$
$E \leftarrow \emptyset$
for $i \leftarrow 1$ to $\tau_{iter}$ do
    $x_{biased} \leftarrow sampleWithRoulette(\mathcal{G}, V, \mathcal{F}')$
    $x_{nearest} \leftarrow nearest(\mathcal{G}, V, x_{biased})$
    $x_{new} \leftarrow steer(x_{nearest}, x_{biased})$
    if $obstacleFree(x_{nearest}, x_{new})$ then
        $V \leftarrow V \cup \{x_{new}\}$
        $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$
    end
end
$\mathcal{G} = (V, E)$
$p_{global} \leftarrow extractPath(\mathcal{G})$
return $p_{global}$
```

Algorithm 5: RRT roulette sampling $(\mathcal{G}, V, \mathcal{F}')$

```
$V' \leftarrow \mathcal{G}_v - V$                         ▷ Get the set of free nodes
$r \leftarrow randomUniform(0, 1)$
if $|V'| > 0$ and $r <= \tau_{random}$ then
    return $randomChoice(V')$                  ▷ Return a random free node
else
    return $weightedChoice(\mathcal{F}')$       ▷ Random weighted frontier selection
end
```

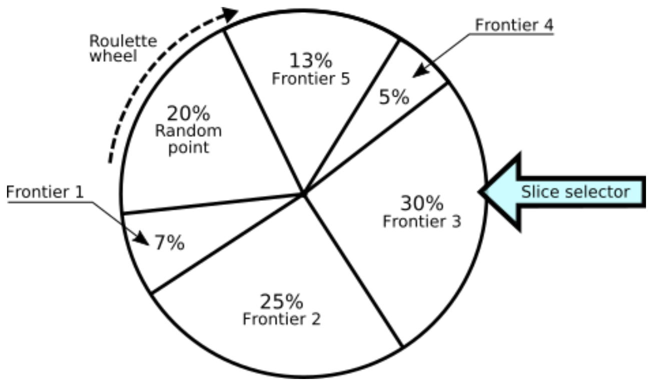


Figure 5.7: Roulette point selection using the normalized information gain as weights. The likelihood of selecting a frontier is given by its expected amount of information gained, represented by the slices at the roulette. More significant information gains represent larger slices and vice-versa. A fixed probability τ_{random} of random frontier selection, e.g. 20%, allows for scaping from local minima.

collisions and should be safe to traverse in a static environment (at least at the time window when the planning took place). Nevertheless, here, the global path is used as a reference for the local planner. This way, the paths generated by the local planner are the ones used for robot navigation, allowing several dynamic advantages for real-world exploration.

5.2.2 RRT local planner with viewpoint filtering

A high level diagram of the local planning process can be observed in Figure 5.8. For local planning, we also use the RRT algorithm, but instead of the complete map of the environment, the RRT operates over a minor point cloud representing the local surroundings of the robot. This local point cloud extraction uses the method proposed in Katz et al. [2007], which approximates the visibility of a point cloud from a given view (in our case, robot pose) without surface reconstruction or normal estimation. The method efficiently removes points that the robot cannot view in a given location, such as occluded by walls, obstacles, or other robots. An example of the point-cloud filtering using the robot's viewpoint can be observed in Figure 5.9, where the subject at the right (a person) is occluded by the first object (dog), and is correctly removed given the viewpoint of the camera.

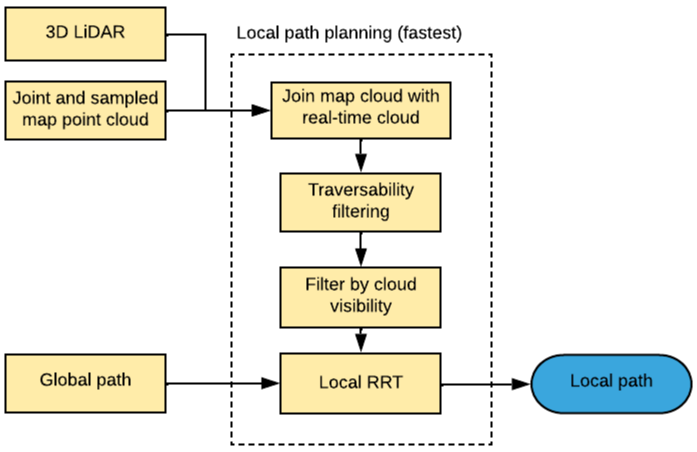
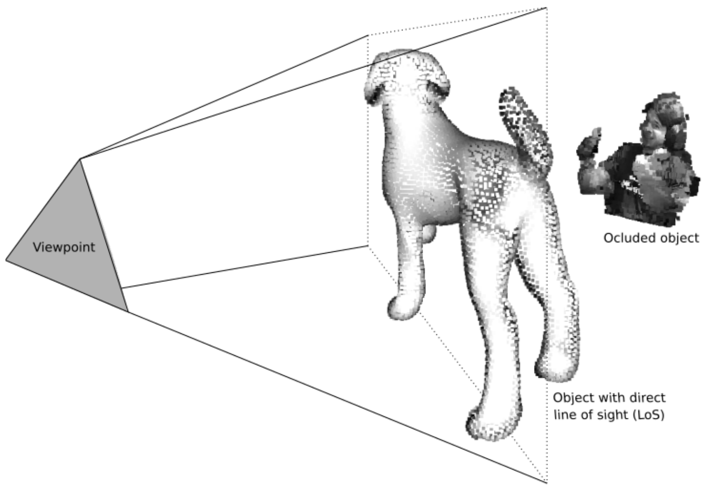
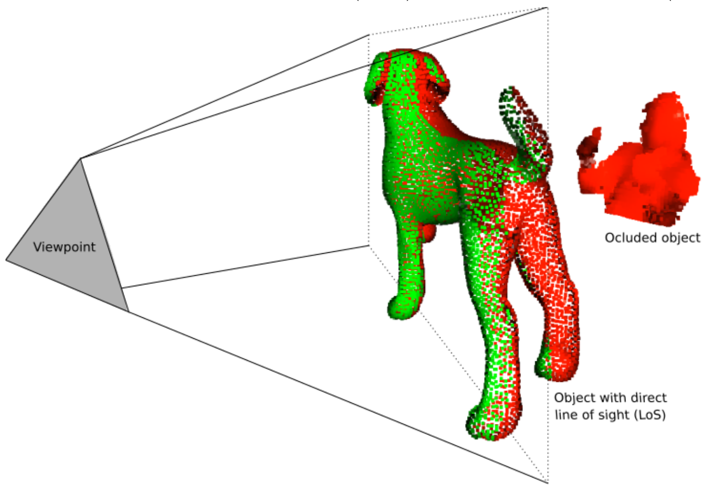

3D LiDAR
Joint and sampled map point cloud
Local path planning (fastest)
Join map cloud with real-time cloud
Traversability filtering
Filter by cloud visibility
Global path
Local RRT
Local path

Figure 5.8: Diagram of the local path planning process (Expanded from Figure 5.2).

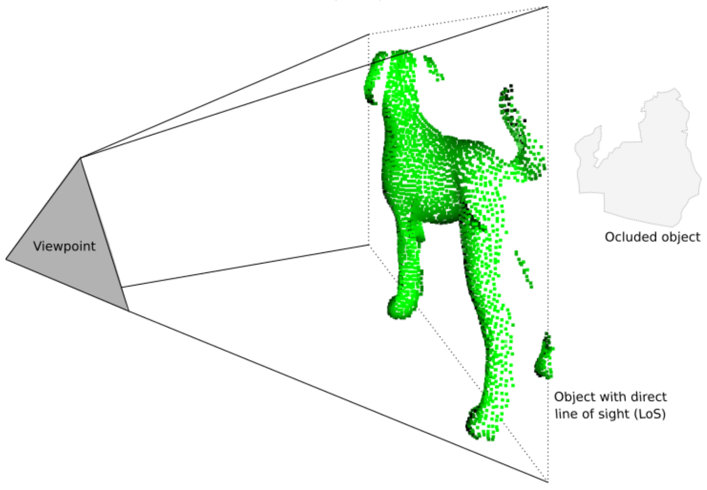
The visibility filtering is described in Algorithm 6, where $\hat{\mathcal{R}}$ is the pose of the robot in the point cloud, $\hat{\mathcal{M}}$ the sampled point cloud of the environment joined with the latest LiDAR raw sensor reading, and τ_{radius} is the threshold used to limit the reach of the filtering. We prioritize using the raw sensor readings since we can explore the faster acquisition rate to detect dynamic obstacles better than waiting for a complete SLAM iteration, which can take seconds to update the point cloud map.



(a) Initial stage with one object (dog) occluding another (person).



(b) Viewpoint analysis signaling the points with direct line of sight (green) and the occluded points (red).



(c) Final stage with the occluded points removed and only the visible part of the dog remains (green).

Figure 5.9: Stages of the point-cloud filtering process using the robot's viewpoint. The filtered cloud with only the points in the direct line of sight of the robot allows for safe local planning, avoiding obstacles.

We added an extra clustering step to the original viewpoint filtering algorithm proposed in Katz et al. [2007] to retain only the points that are reachable by the robot, such as ground areas. The τ_{radius} constant is not straightforward to select since it behaves differently in sparse and dense point clouds, therefore it must be selected via experimentation per robot configuration. The resulting viewpoint cloud is also filtered by angle using Algorithm 3.

Algorithm 6: Visibility filtering of point clouds ($\hat{\mathcal{R}}, \hat{\mathcal{M}}, \tau_{radius}$)

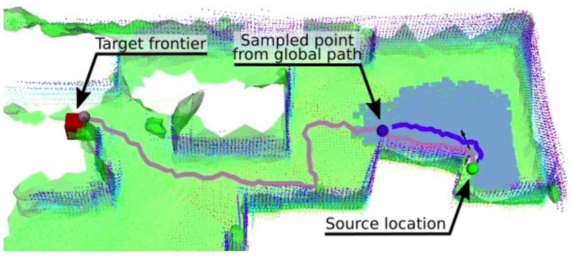
```
    $\mathcal{E} \leftarrow sphericalProjection(\hat{\mathcal{M}}, \hat{\mathcal{R}}, \tau_{radius})$
    $\mathcal{H} \leftarrow convexHull(\mathcal{E}, \hat{\mathcal{R}})$
    $\hat{\mathcal{M}} \leftarrow extractPoints(\mathcal{H}, \hat{\mathcal{R}})$    ▷ Points in the hull are the visible ones
    $\mathcal{C} \leftarrow DBSCAN(\hat{\mathcal{M}}, eps, \mathcal{C}_{min})$
    $\hat{\mathcal{M}} \leftarrow filterClusterByOrigin(\mathcal{C}, \hat{\mathcal{R}})$    ▷ Remove unreachable points
    return $\hat{\mathcal{M}}$
```

The path navigation is performed by sampling the global path p_{global} in a lookahead fashion – similar to the way a pure pursuit algorithm estimates the new point to visit [Coulter, 1992]. We sample p_{global} using a sphere of radius τ_{radius} from the robot position, and get the point inside this radius closer to the target frontier ($x_{sampled}$). This way, the robot is chasing a "constantly moving point" over the global path that is at some distance ahead of it. The RRT planner then constantly estimates the path from the current robot position to the closest point to $x_{sampled}$ in $\hat{\mathcal{M}}$. For controlling the robot motion, we use the vector field low-level controller presented in Section 3.2.5. This local planning and navigation process is shown in Figure 5.10 (online video available*).

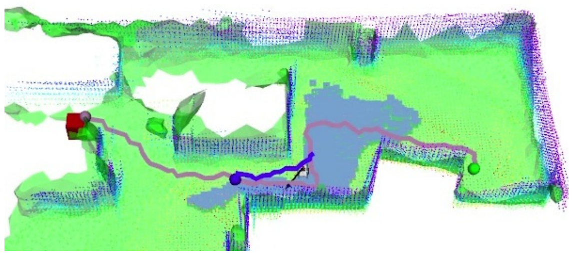
Using a local point cloud is critical for planning safe paths with reachable regions and detecting when the global path cannot be followed anymore. A typical example of this is when the global path is interrupted by an obstacle. In this case, if the distance of the robot to the end of the local path is less than a threshold τ_{target}, we verify if the global target was reached and, if true, end this exploration cycle. On the other hand, if the threshold was reached but the global target is still far away, we assume that the robot ended the local path without reaching the frontier and needs to replan to another frontier. Algorithm 7 describes the proposed local planning process.

The LeGO-LOAM [Shan and Englot, 2018] algorithm that we use for SLAM produces a point cloud of the environment, and additive changes in the environment are registered with detail; however, subtractive changes in the environment are not handled correctly by LeGO-LOAM. This is because the algorithm does not subtract points

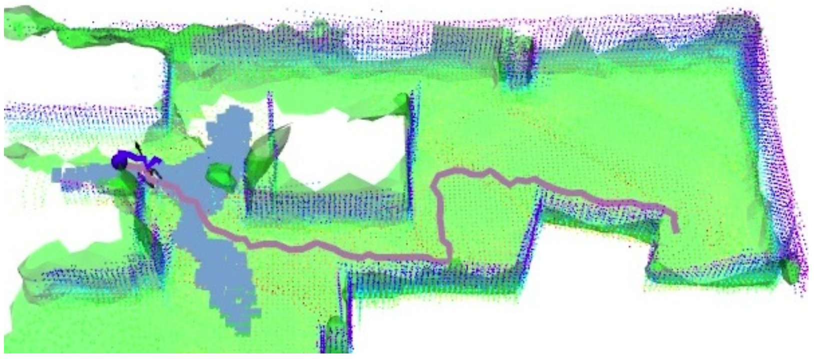
*Local planner navigation video: `https://youtu.be/FN2rdfWwqpE`



(a) The initial stage of planning to navigate into a frontier.



(b) The visible point cloud adapts to the environment (light blue).



(c) The local path gets smaller when reaching a frontier or obstacle (dark blue).

Figure 5.10: Process of using the local planner to navigate towards a frontier using the global path as a reference. The global path is shown in light pink, and the local path in dark blue.

Algorithm 7: Local RRT plan and navigation $(\hat{\mathcal{M}}, \hat{\mathcal{R}}, p_{global})$

```
$x_{goal} \leftarrow p_{global}.lastElement()$
while $|p_{global}| > 2$ do
    $x_{sampled}, p_{global} \leftarrow sampleAndReducePath(p_{global}, \hat{\mathcal{R}}, \tau_{radius})$
    $x_{visit} \leftarrow closestPoint(x_{sampled}, \hat{\mathcal{M}})$
    $p_{local} \leftarrow RRT(x_{visit}, \hat{\mathcal{M}}, \hat{\mathcal{R}})$
    $sendPathToNavigator(p_{local})$
    if $distance(\hat{\mathcal{R}}, x_{visit}) < \tau_{target}$ then
        if $x_{visit} == x_{goal}$ then
            return $True$                                ▷ Global path completed
        else
            return $False$
        end
    end
end
return $False$
```

but only adds new samples. So, for example, moving an object would leave movement traces along the path since the previous object's points will not be removed from the cloud. A way to improve the additive nature of the algorithm is using a probabilistic map (such as Octomap), allowing points to have a probability of occupancy. However, a naively loosely coupled integration of LeGO-LOAM with a probabilistic map could have problems synchronizing loop closures and drastic map updates. Another point of attention is that our current approach uses sparse depth sensors (multi-line LiDAR), making it difficult to remove other sparse objects altogether since there is a reduced chance of the sensor's rays touching all of the point traces left by the movement of the object. The difficulty in matching rays with points could be partially solved by increasing the size of the voxels (thus reducing the resolution to increase the likelihood of a ray collision) or by using other types of sensors, such as short-range dense depth sensors (ToF cameras), but those analyses are left for future investigations.

5.3 Experiments and results

As in the previous set of experiments, all evaluations were executed in virtual environments using the simulated version of the EspeleoRobô [Azpurua et al., 2019; Cid et al., 2020]. For this set of experiments, a newer version of the CoppeliaSim[†] simulator was used (v4.3.0), executed with ROS Noetic under Ubuntu 20.04. The laptop used for the experiments has an Intel i7-10875H CPU @ 2.30GHz, 32GB of RAM, and an NVIDIA RTX 2070 Super GPU with 8GB of VRAM.

We validated the performance of the proposed exploration pipeline using three scenarios: (a) a single-level cave map extracted from the DARPA SubT challenge, (b) a multi-level cave map also from the DARPA challenge, and (c) a synthetic simplified cave map in a flat environment (Figure 4.3). As before, the EspeleoRobô robot was equipped with a simulated Ouster OS1 LiDAR sensor and an IMU. For all experiments, the maximum slope is defined as $\theta_{max} = 30°$.

5.3.1 RRT global path planning

The RRT global planner allows for a rapid plan without using a hardcoded method for target definition by using biases according to the expected information from visiting the frontiers. A sequence of RRT global paths can be observed in Figure 5.11. The paths are shown using a 2D top-view projection of the 'Single-level" DARPA cave, a complete three-dimensional map with slopes and obstacles. The results show the growth

[†]`http://www.coppeliarobotics.com`

of the RRT tree over the point cloud, generating feasible paths that tend towards the most informative frontier and considering the cost of traversing the map. The red dots represent the reachable frontiers, the blue dot the robot position, the RRT tree is shown in orange, and the selected branch into the target frontier is shown in light purple (online video available[‡]).

With this method, the robot will not always visit a close frontier since the probabilistic nature of the algorithm will bias toward other more informative areas, maximizing the global exploration rate instead of greedily selecting the closest frontiers.

5.3.2 Adaptative obstacle avoidance

One of the key benefits of having a local/global planner is the capacity to change the local plan to avoid immediate dangers and obstacles or to detect unreachable goals, preventing the robot from going into a risk zone. In this sense, we validated the RRT path planner for obstacle avoidance in three experiments: (i) a corridor-like environment with a complete blockade during the mission execution, (ii) a corridor-like environment with a partial blockade, and (iii) a cave-like "T" environment with two exploration possibilities, with one of them completely blocked during the mission execution.

The corridor-like environment is shown in Figure 5.12, where the robot spawns at area “A” and needs to explore area “B”. In the middle of the exploration mission, an obstacle blocks the entire corridor (red arrow), and the robot stops without colliding with the blockade. As seen in Figure 5.12b, the visibility point cloud is reduced when the obstacle appears, and the robot’s best local plan has a length less than τ_{target}, making the robot stop (online video available[§]).

In the second experiment (Figure 5.13), an obstacle appears over the global path, forcing the local planner to contour the object to avoid a collision and reach the target destination. In this experiment, it is possible to observe the visibility cloud’s behavior, which decreases in size and changes its spatial topology, adapting itself to the environmental changes (online video available[¶]).

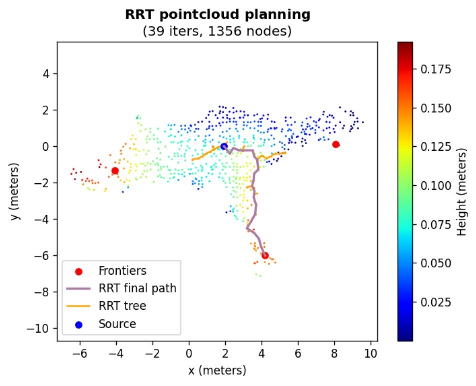
The third experiment with obstacles, which is performed in a larger scenario with two bifurcations, can be observed in Figure 5.14. In this scenario, the robot spawns in the middle of the environment and tries to explore area “A”. Mid-mission into area “A” a blockade appears, and the robot continues the exploration into area “B” which is free of blockades (online video available[‖]).

[‡]MI-RRT exploration iterative process (video): `https://youtu.be/DKa8EVRWNzk`

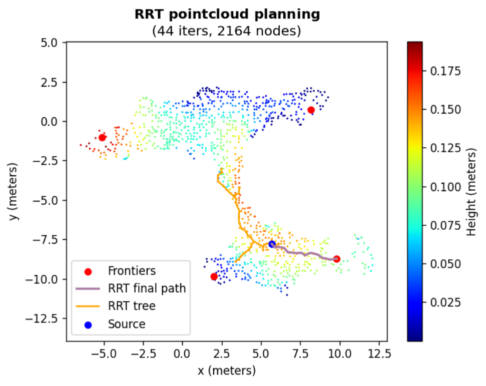
[§]Exploration in a dead-end corridor (video): `https://youtu.be/p04KAPwjiHg`

[¶]Obstacle avoidance in a corridor-like environment (video): `https://youtu.be/aZu_y8-CS9A`

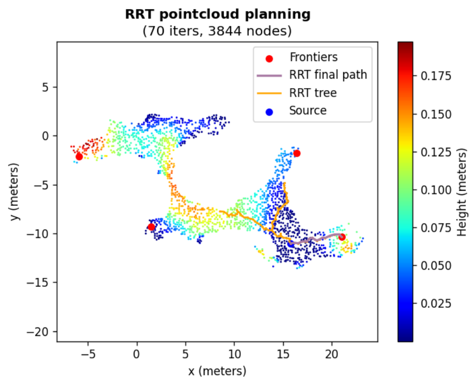
[‖]Obstacle avoidance in a bifurcated environment (video): `https://youtu.be/d6wrfBwOjv0`



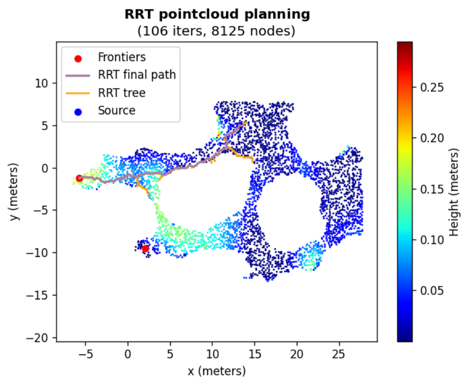
(a)



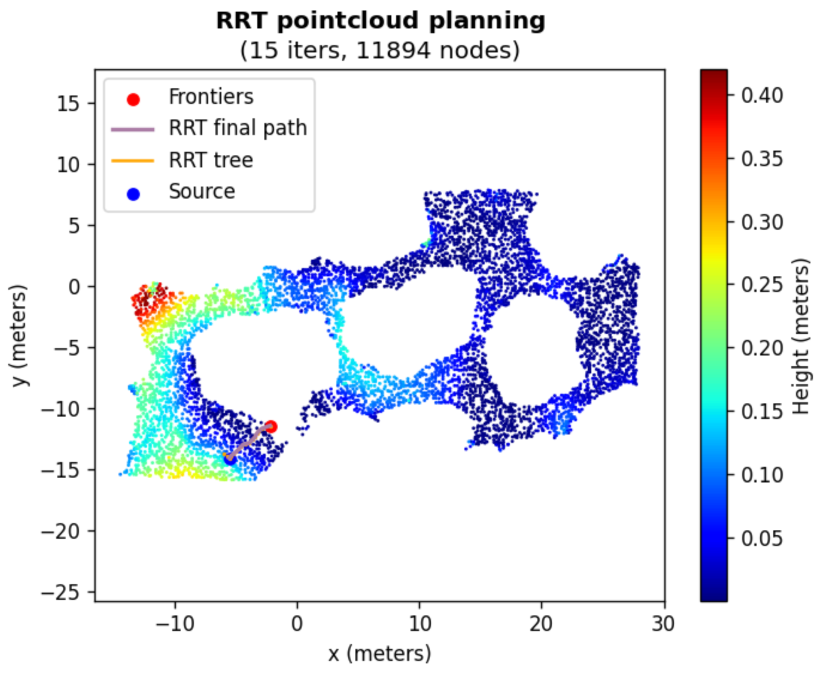
(b)



(c)

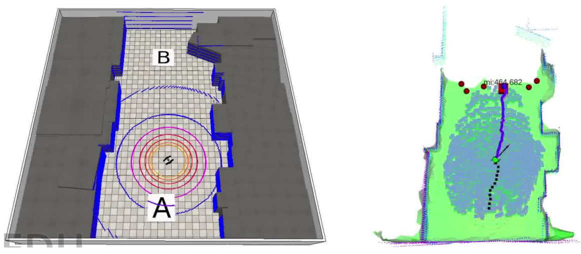


(d)

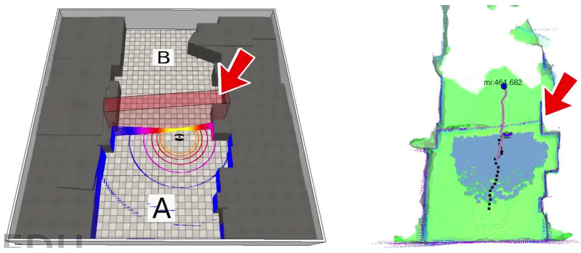


(e) Final exploration state.

Figure 5.11: RRT paths generated over an exploration mission at the “Single-level” cave scenario (a-e). The orange paths are the branches of the RRT tree, and the light purple paths are the selected branch that leads the robot (blue dot) to one of the frontiers (red dots).



(a) Initial stage of a corridor without obstacles. The robot started at area "A".



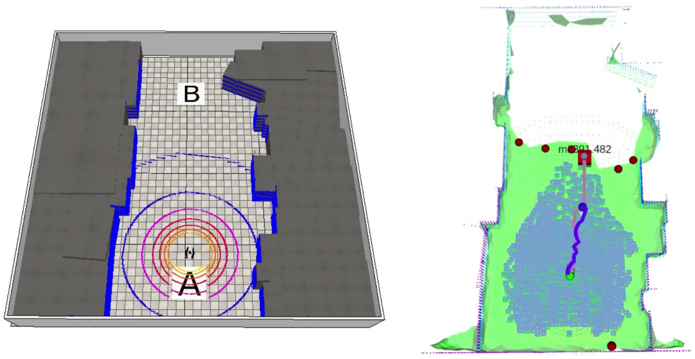
(b) An obstacle prevents the robot from performing the global path (red arrows), showing reduced visibility, and forcing the robot to stop the exploration.

Figure 5.12: Blockade avoidance in a corridor-like environment: the robot spawns in the area "A" and needs to explore area "B", while in mid-mission, a complete blockade appears in the environment. At the left is the simulated environment, and at the right is the reconstructed map made online by the robot. The reconstructed mesh is shown in green, the local path in dark blue, the global path in light pink, and the point cloud visible by the robot in light blue.

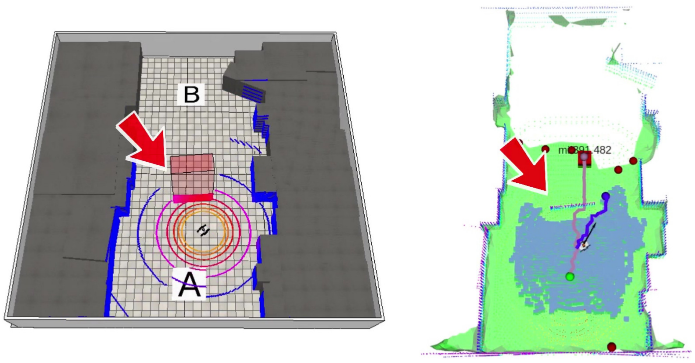
5.3.3 Path planning performance

These experiments compare the performance of the exact algorithm and the proposed global path planner using the biased RRT. In the exact version of the planner (Chapter 4), the planning step is performed after the mesh reconstruction, when the algorithm transforms the mesh into a graph, and the Dijkstra search algorithm is used to estimate the best paths for each frontier. On the other hand, the biased RRT performs the frontier selection and the path planning simultaneously, improving performance.

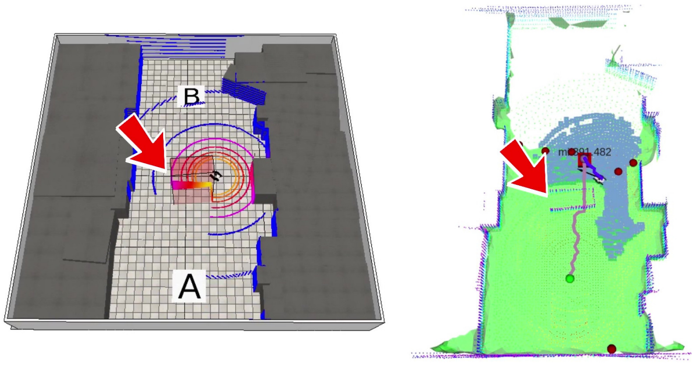
Figure 5.15 shows the time spent planning in different situations. In Figure 5.15a the average times with the confidence intervals showed that for the majority of cases, the RRT planner performed better than the exact version. The results in Figure 5.15b showed that the time spent in planning is closely related to the number of available frontiers to explore. This could be explained by the fact that the exact method will need to run the Dijkstra algorithm for every frontier to extract the correct metric, while the RRT-based algorithm only needs to run once. Also, the MI-RRT computing time seems to decrease with an increased number of frontiers. This decrease in time could be explained by the fact that in the scenarios validated in this thesis, with more



(a) Initial stage of a corridor without obstacles. The robot started at area “A”.

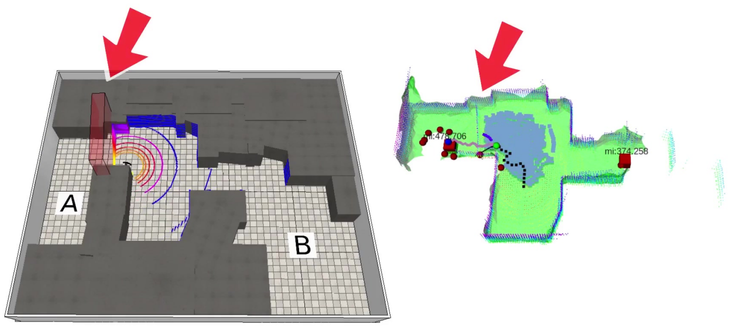


(b) An obstacle appears over the global path (red arrows), making the robot to turn around to prevent a collision.

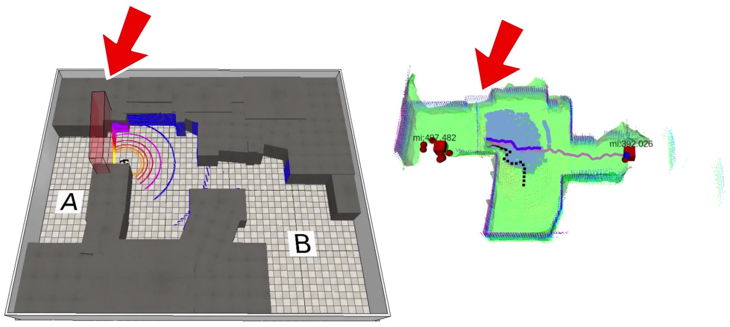


(c) The robot contours the obstacle and reach the target position.

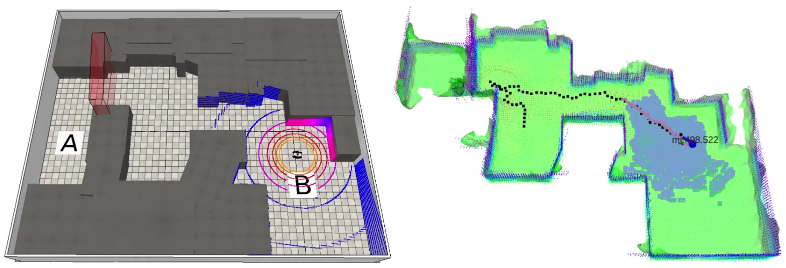
Figure 5.13: Obstacle avoidance in a corridor-like environment: the robot spawns in the area “A” and needs to explore area “B”, while in mid-mission, a partial blockade appears over the global path. At the left is the simulated environment, and at the right is the reconstructed map made online by the robot. The reconstructed mesh is shown in green, the local path in dark blue, the global path in light pink, and the point cloud visible by the robot in light blue.



(a) An obstacle prevents the robot from performing the global path into area “A” (red arrows), showing reduced visibility, and forcing the robot to change the target frontier to area “B”.



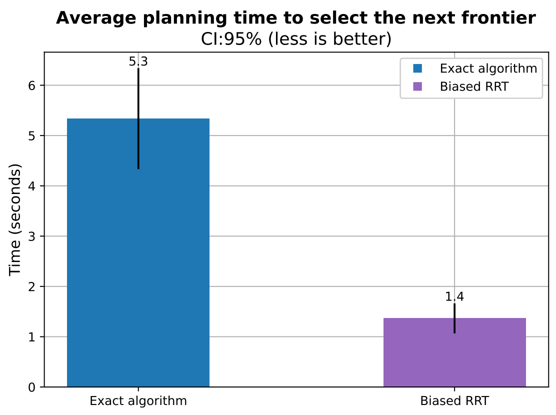
(b) The robot generates a global path to area “B”.



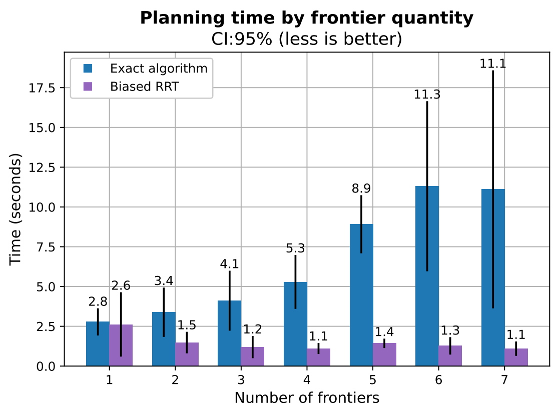
(c) Area “B” is completely visited.

Figure 5.14: Obstacle avoidance in an environment with bifurcations: the robot spawns in the middle of the map and needs to explore areas “A” and “B”. The path to “A” is blocked during the mission, forcing the robot to continue exploring area “B”. The reconstructed mesh is shown in green, the local path in dark blue, the global path in light pink, and the point cloud visible by the robot in light blue.

available frontiers, there is a greater chance of a frontier closer to the robot position, and with a small number of frontiers, there are more chances of the frontier beign far away, requiring the use of more nodes to reach them.



(a) Average time to select the next frontier (global planner).



(b) Average time to select a frontier, grouped per frontier's quantity.

Figure 5.15: Planning times for the exact and RRT based algorithms: (a) the total average times between the RRT and exact planners, and (b) a comparison in times by the number of available frontiers. The analysis was performed over 100 runs.

5.3.4 Complete autonomous exploration

The RRT planner was validated with the best and worst frontier selections of the previous exact exploration method shown in Chapter 4: (i) the proposed approach using the information gain and traversability path generation of the frontiers to select the next visit point, and (ii) the random selection of the frontier using the exact terrain-aware path. The results of this analysis can be observed in Fig. 5.16, where the displayed lines are the mean RMSE for every timestep of five repetitions per experiment. The Equation 4.5 was used for the RMSE comparison between point clouds. The standard deviation (σ) in the RRT metric is also shown in light purple. The timesteps were estimated using only the time the robot moved and did not include the paths' computing time. The RRT metric (purple) has a better performance than the orange metric (exact path planning with random frontier selection). However, the exact metric for frontier selection (MI ratio) still performs better than the others. The probabilistic nature of the RRT with bias could explain the inferior performance of the exact metric since there is no guarantee that the best path is selected at all times (online video available**).

**MI-RRT Exploration examples (video): `https://youtu.be/zivjb9ovugQ` and `https://youtu.be/0tboyqh6GWI`

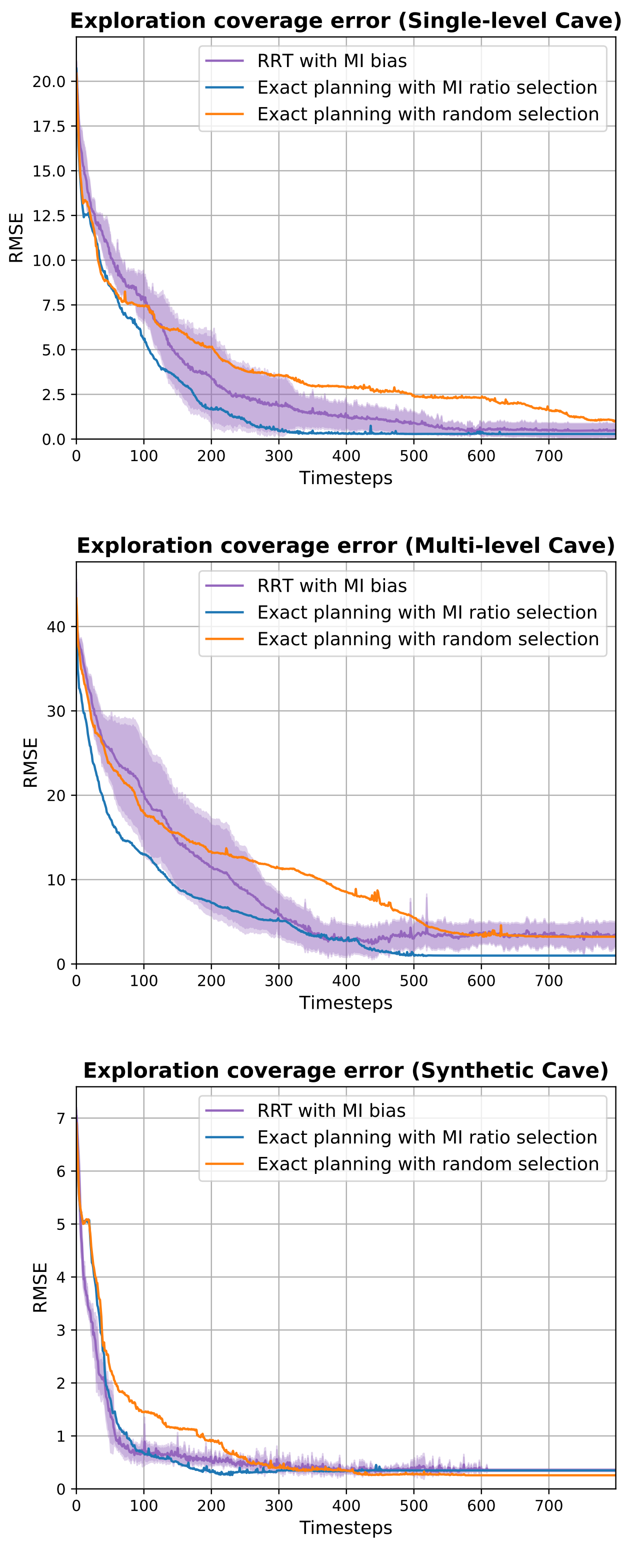
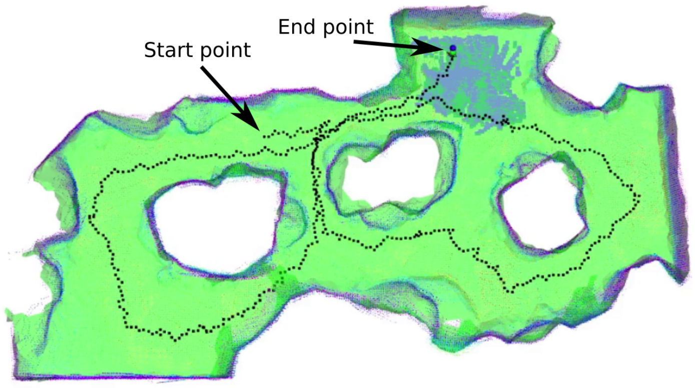
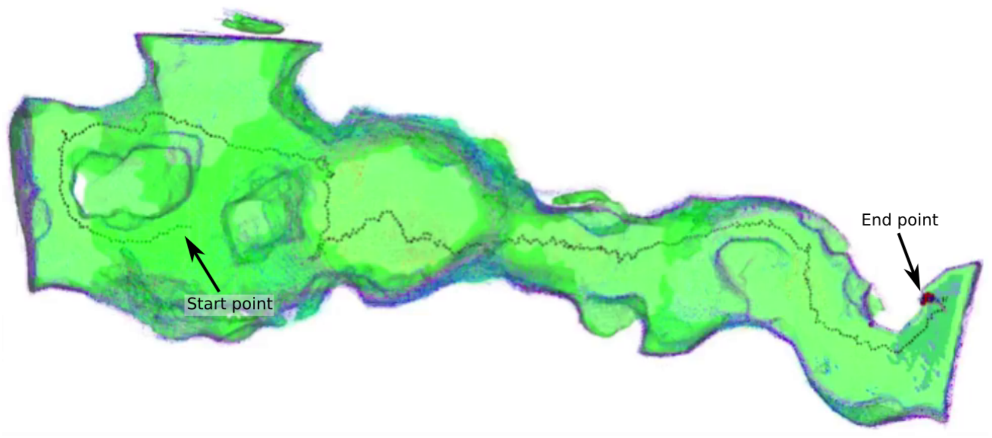


Figure 5.16: RRT exploration error (RMSE) mean of five runs comparing the real-time SLAM point cloud with a reference map for the: (a) single-level cave, (b) multi-level cave, and (c) synthetic cave.

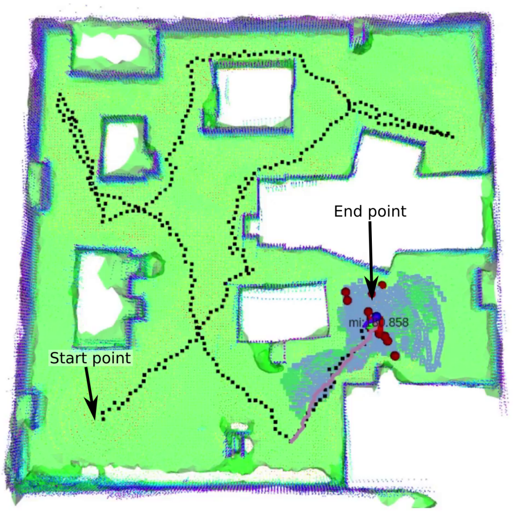
The paths performed by the robot when using the proposed strategy with the RRT algorithm can be observed in Figure 5.17. As with the exact method with MI ratio frontier selection, the robot's exploratory decisions lead to mostly non-repetitive paths where the already covered areas are less likely to be revisited.



(a) Single-level cave.



(b) Multi-level cave



(c) Synthetic cave map.

Figure 5.17: Exploration paths generated by the proposed RRT exploration strategy for the: (a) single-level cave, (b) multi-level cave, and (c) synthetic cave. The black dots shows the robot's LiDAR odometry along the exploration mission.

Chapter 6

Conclusion and Future Work

"Now this is not the end.
It is not even the beginning of the end.
But it is, perhaps, the end of the beginning."
— Wiston Churchill.

This work has presented a pipeline for three-dimensional path planning for ground robots in rugged terrains and two incremental proposals for terrain-aware autonomous exploration for confined spaces. In this sense, we proposed a deterministic exploration method using optimal path search algorithms and a probabilistic method using a biased RRT that we call MI-RRT. The basis of our exploration proposals uses a novel approach that combines the cost of traversing rugged terrains and the expected information gathered by visiting a frontier. Our proposed method uses octrees, meshes, and the raw point cloud to calculate the most informative frontiers and generate safe paths considering terrain traversability, distance, and energy consumption. Furthermore, unlike traditional exploration methods, the proposed method works in complex 3D environments without assuming any priors over the map structure.

6.1 Conclusion

Throughout this dissertation, we have answered three main questions: "How to reach a goal?" (planning and navigation), "How to select the next best objective?" (exploration) and "How to plan faster?" (faster path planning for exploration). The answers to these questions allow a robot to explore the environment autonomously, safely, and efficiently.

In contrast to other start-of-the-art works on real-world exploration in confined scenarios, in this work, we proposed using a limited sensorial suite composed of only one

3D LiDAR, an IMU, and wheel encoders. This limited suite was shown as sufficient for terrain-aware exploration, being more economic, lightweight and consuming less energy than other approaches, with only a few drawbacks. For example, the lack of a dense sensor, such as a projected depth camera, makes detecting small untraversable areas at the proximities of the robot a challenge since our 3D sensor is sparse. Of course, this is a concern for an actual robot deployment; however, in our tests, we did not observe dangerous situations generated by this condition.

We learned via real and simulated experimentation that having a robust code-base and integration tests through the entire exploration pipeline is critical for robust robot autonomy. Test integration is something commonly overlooked in robotic developments. While we did not address this specific point in the theoretical aspects of the dissertation, a test-driven development of the software implementation allowed us to integrate multiple systems quicker, with fewer operational failures. Also, a considerable amount of effort was spent making the pipeline efficient enough to run online in the embedded robot computer. Since the networking infrastructure inside confined spaces is generally unreliable, an autonomous exploration robot must make most decisions by itself, using local resources only.

In this dissertation, we presented incremental solutions to different navigation and exploration problems. The two proposed exploration approaches suit different situations and can work interchangeably. For example, the deterministic pipeline is more appropriate when the best possible outcome is desired, disregarding CPU consumption or time. On the other hand, the probabilistic approach is more suitable for less powerful setups or when processing time is more important than the optimal solution or the environment has dynamic obstacles.

Regarding path planning and navigation in rugged terrains, in Chapter 3, we proposed a method to safely generate paths using multiple terrain-aware metrics. A linear combination of those metrics compounds the edge weights between the nodes of a traversability graph extracted from the 3D mesh reconstruction of the environment. An exact path planning algorithm such as Dijkstra estimates the path with the minimum cost. Simulated and real-world experiments in a colonial iron-ore mine show the technique's feasibility. However, despite the great representation capabilities of meshes, generating them could be a CPU-intensive task for noisy or massive point clouds.

Our autonomous deterministic exploration method shown in Chapter 4 is based on the expected information gain a frontier will bring after visiting it and uses the previous navigation method to estimate the cost of visiting frontiers. The information gain of a frontier is estimated by calculating the Shannon Entropy by ray tracing the FoV of the 3D LiDAR sensor over a frontier location. In our first exploration proposal, which we call “exact” or “deterministic”, the best-next place to visit uses a ratio between the expected information gain of a frontier by the navigation cost of actually visiting it. The proposed pipeline works for outdoor and confined scenarios since it does not need external localization. The mutual information metric shows increased performance in more comprehensive environments with many possible visiting locations and obstacles. In this sense, the algorithm could detect frontiers with small informational rewards and prevent the robot from wasting time and energy visiting them. In small scenarios, the benefit of selecting a frontier only by information and traversability decreases, and it is similar to the most straightforward nearest frontier selection.

Finally, in Chapter 5, we presented a faster exploration method using a biased RRT algorithm that we call MI-RRT, capable of global/local planning and obstacle avoidance using the raw point cloud. This method is built upon the previous exploration algorithm presented in Chapter 3, though it is more computationally efficient than the deterministic exploration by simultaneously allowing for global path planning and frontier selection without needing expensive multiple path computations. Furthermore, the method is probabilistic and can be executed online at the cost of a slight decrease in exploration performance, as shown in the simulated experiments. The local planner using this approach can estimate multiple paths per second, allowing for reactive obstacle avoidance when following the global path

As stated before, the proposed method’s main disadvantages are CPU-intensive operations when reconstructing the mesh. Even though having a full mesh facilitates many path planning tasks, including straightforwardly removing many untraversable areas or estimating the reachable frontiers, the process of mesh generation is prone to outliers –typical in real-world data– as shown in the real indoor navigation experiments. When the map point cloud is noisy, preliminary filtering steps are needed to reduce the final mesh error and processing time. The proposed method for exploration is currently dependent on the LiDAR sensor intrinsic parameters only, which could be significantly improved with some terrain reconstruction methods over the frontiers to infer what a continuation of the frontier would be, therefore improving the expected lidar virtual measurements’ with a more realistic ray-casting. The problem of generating the mesh is decreased in the probabilistic method since it is used the least amount of times possible, performing the local and global planning directly in the point cloud.

All implemented algorithms were published as an open-source package and available online for free at `https://github.com/verlab/terrain_aware_exploration`.

6.2 Future Work

The proposed exploration pipeline leads to numerous open questions and research directions, especially when dealing with multi-robot or learn-based approaches. In the following, we highlight some of the most natural evolutions of the proposed method that can be performed in the short-mid term:

- Evaluate the performance of the other path planning algorithms such as Dijkstra or A* for local planning instead of RRT;
- Evaluate D* for global planning;
- Improve the rewiring step of the RRT algorithm to convert it into an RRT*, warranting an optimal bound;
- Segment the map point cloud in regions and use those regions for sampling instead of the direct frontier location when using the biased MI-RRT algorithm;
- Update the metrics weights dynamically depending on the robot's state and the environment. For example, focus on using less energy when going back to the base, favor shortest paths when reporting the location of an injured person, or reduce energy consumption when the battery has a low charge;
- Add ground friction, mud, or other terrain characteristics to the graph weights;
- Since we are generating a graph of the environment, we could also add semantic or topological information to this map to allow more complex exploration behaviors;
- Use dynamic velocity changes according to the terrain characteristics and edge weights.

On a broader research timeline, it is possible to study time-bounded exploration. Exploration with limited time windows will certainly generate different interesting behaviors since the robot must choose frontiers and terrain weights considering that not all frontiers can be visited. Other future works, for example, can improve the map information estimation using the topology of known frontier borders to learn realistic 3D map expansions instead of using the sensor model alone. Also, since the meshing algorithm is the bottleneck in this pipeline, looking into minimizing the computational cost of this operation is key to allowing faster exploration missions.

Another critical area for future work is cooperation. Multiple robots could improve the efficiency and resiliency of exploratory robotic behaviors. In this regard, one possible next goal can be homogeneous terrestrial cooperation, where a group of centralized or decentralized robots performs the exploration without repeating paths between them, generating a comprehensive shared map combining the group's gathered information. However, cooperation is a challenge by itself, and it will need the research of task allocation methods appropriate for confined scenarios using robots with limited sensing and communication range to enable efficient multi-robot exploration behaviors.

The natural step after homogeneous cooperation is heterogeneous collaboration. Heterogeneous exploration teams could perform aerial/terrestrial collaboration, allowing the privileged mobility of aerial platforms could be enhanced by the extended range of their grounded counterparts. Following this idea, a possible efficient combination for exploration can use terrestrial and aerial platforms, where the terrestrial robots are used as carriers for smaller and lightweight aerial robots. The aerial robots will engage when the terrestrial platform encounters a non-traversable area or obstacle, and after performing a local exploration, the drone will go back to the carrier for charging. As the path's cost generated by the proposed techniques depends on particular robot characteristics such as maximum slope angle and maximum traversable obstacle height, heterogeneous terrestrial teams could inspect different parts of the map by defining different mapping constants. When dealing with multi-robot teams, localization and communication are critical challenges that should also be addressed.

Bibliography

Agha, A., Otsu, K., Morrell, B., Fan, D. D., Thakker, R., Santamaria-Navarro, A., Kim, S.-K., Bouman, A., Lei, X., Edlund, J., et al. (2021). Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *arXiv preprint arXiv:2103.11470.*

Ahmad, S., Mills, A. B., Rush, E. R., Frew, E. W., and Humbert, J. S. (2021). 3d reactive control and frontier-based exploration for unstructured environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2289--2296. IEEE.

Akbarally, H. and Kleeman, L. (1995). A sonar sensor for accurate 3d target localisation and classification. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 3003--3008. IEEE.

Akbari, A., Chhabra, P., Bhandari, U., and Bernardini, S. (2020). Intelligent exploration and autonomous navigation in confined spaces. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA*, pages 25--29.

Alfeld, P. (1984). A trivariate clough—tocher scheme for tetrahedral data. *Computer Aided Geometric Design*, 1(2):169--181.

Azpúrua, H., Campos, M. F., and Macharet, D. G. (2021a). Three-dimensional terrain aware autonomous exploration for subterranean and confined spaces. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2443--2449. IEEE.

Azpúrua, H., Rezende, A., Potje, G., da Cruz Júnior, G. P., Fernandes, R., Miranda, V., de Resende Filho, L. W., Domingues, J., Rocha, F., de Sousa, F. L. M., et al. (2021b). Towards semi-autonomous robotic inspection and mapping in confined spaces with the espeleorobô. *Journal of Intelligent & Robotic Systems*, 101(4):1--27.

Azpurua, H., Rocha, F., Garcia, G., Santos, A. S., Cota, E., Barros, L. G., Thiago, A. S., Pessin, G., and Freitas, G. M. (2019). Espeleorobô-a robotic device to inspect confined environments. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 17--23. IEEE.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469--483.

Bayer, J. and Faigl, J. (2019). On autonomous spatial exploration with small hexapod walking robot using tracking camera intel realsense t265. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1--6. IEEE.

Bayer, Janand Faigl, J. (2020). Speeded up elevation map for exploration of large-scale subterranean environments. In Mazal, J., Fagiolini, A., and Vasik, P., editors, Modelling and Simulation for Autonomous Systems, pages 190--202, Cham. Springer International Publishing.

Besselmann, M. G., Puck, L., Steffen, L., Roennau, A., and Dillmann, R. (2021). Vdb-mapping: A high resolution and real-time capable 3d mapping framework for versatile mobile robots. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 448--454. IEEE.

Bianco, S., Ciocca, G., and Marelli, D. (2018). Evaluating the performance of structure from motion pipelines. *Journal of Imaging*, 4(8):98.

Bijelic, M., Gruber, T., and Ritter, W. (2018). A benchmark for lidar sensors in fog: Is detection breaking down? In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 760--767. IEEE.

Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2018). Receding horizon path planning for 3d exploration and surface inspection. *Autonomous Robots*, 42(2):291--306.

Bissmarck, F., Svensson, M., and Tolt, G. (2015). Efficient algorithms for next best view evaluation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5876--5883. IEEE.

Blochliger, F., Fehr, M., Dymczyk, M., Schneider, T., and Siegwart, R. (2018). Topomap: Topological mapping and navigation based on visual slam maps. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1--9. IEEE.

Bouman, A., Ginting, M. F., Alatur, N., Palieri, M., Fan, D. D., Touma, T., Pailevanian, T., Kim, S.-K., Otsu, K., Burdick, J., et al. (2020). Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion. *arXiv preprint arXiv:2010.09259*.

Bradsher, K. (2016). Taiwan earthquake investigators arrest developer of collapsed building.

Brossard, M. and Bonnabel, S. (2019). Learning wheel odometry and imu errors for localization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 291--297. IEEE.

Buchanan, R., Bandyopadhyay, T., Bjelonic, M., Wellhausen, L., Hutter, M., and Kottege, N. (2019). Walking posture adaptation for legged robot navigation in confined spaces. *IEEE Robotics and Automation Letters*, 4(2):2148--2155.

Buchanan, R., Wellhausen, L., Bjelonic, M., Bandyopadhyay, T., Kottege, N., and Hutter, M. (2021). Perceptive whole-body planning for multilegged robots in confined spaces. *Journal of Field Robotics*, 38(1):68--84.

Chang, K.-H. (2015). Chapter 5 - multiobjective optimization and advanced topics. In Chang, K.-H., editor, Design Theory and Methods Using CAD/CAE, pages 325–406. Academic Press, Boston.

Choset, H. and Nagatani, K. (2001). Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on robotics and automation*, 17(2):125--137.

Cid, A., Nazário, M., Sathler, M., Martins, F., Domingues, J., Delunardo, M., Alves, P., Teotônio, R., Barros, L. G., Rezende, A., et al. (2020). A simulated environment for the development and validation of an inspection robot for confined spaces. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1--6. IEEE.

Cieslewski, T., Kaufmann, E., and Scaramuzza, D. (2017). Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2135--2142. IEEE.

Coello, C. A. (2000). An updated survey of ga-based multiobjective optimization techniques. *ACM Computing Surveys (CSUR)*, 32(2):109--143.

Connolly, C. (1985). The determination of next best views. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 432--435. IEEE.

Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

Coumans, E. and Bai, Y. (2016–2019). Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`.

da Cruz Júnior, G. P., do Carmo Matos, L. V., Azpúrua, H., Pessin, G., and Freitas, G. M. (2020). Investigação de técnicas lidar slam para um dispositivo robótico de inspeção de ambientes confinados. *Anais da Sociedade Brasileira de Automática*, 2(1).

Dai, A., Papatheodorou, S., Funk, N., Tzoumanikas, D., and Leutenegger, S. (2020). Fast frontier-based information-driven autonomous exploration with an mav. *arXiv preprint arXiv:2002.04440*.

Dal Mutto, C., Zanuttigh, P., and Cortelazzo, G. M. (2012). *Time-of-flight cameras and Microsoft KinectTM*. Springer Science & Business Media.

Dang, T., Khattak, S., Mascarich, F., and Alexis, K. (2019). Explore locally, plan globally: A path planning framework for autonomous robotic exploration in subterranean environments. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 9--16. IEEE.

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*, 37(8):1363--1388.

DARPA (2019). Darpa subterranean challenge tunnel circuit environment preview.

DARPA (2020). Darpa subterranean challenge cave environment preview.

Delmerico, J., Isler, S., Sabzevari, R., and Scaramuzza, D. (2018). A comparison of volumetric information gain metrics for active 3d object reconstruction. *Autonomous Robots*, 42(2):197--208.

Dharmadhikari, M., Dang, T., Solanka, L., Loje, J., Nguyen, H., Khedekar, N., and Alexis, K. (2020). Motion primitives-based path planning for fast and agile exploration using aerial robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 179--185. IEEE.

Dickmann, J., Appenrodt, N., Bloecher, H.-L., Brenk, C., Hackbarth, T., Hahn, M., Klappstein, J., Muntzinger, M., and Sailer, A. (2014). Radar contribution to highly automated driving. In *2014 44th European Microwave Conference*, pages 1715--1718. IEEE.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269--271.

Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., and Jurišica, L. (2014). Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59--69.

Ebadi, K., Chang, Y., Palieri, M., Stephens, A., Hatteland, A., Heiden, E., Thakur, A., Funabiki, N., Morrell, B., Wood, S., et al. (2020). Lamp: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 80--86. IEEE.

Fairfield, N., Kantor, G., and Wettergreen, D. (2006). Towards particle filter slam with three dimensional evidence grids in a flooded subterranean environment. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 3575--3580. IEEE.

Fankhauser, P. and Hutter, M. (2016). A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In Koubaa, A., editor, Robot Operating System (ROS) – The Complete Reference (Volume 1), chapter 5. Springer.

Fankhauser, P. and Hutter, M. (2018). Anymal: a unique quadruped robot conquering harsh environments. *Research Features*, 1(126):54--57.

Frazzoli, E., Dahleh, M. A., and Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *Journal of guidance, control, and dynamics*, 25(1):116--129.

Freitas, G., Gleizer, G., Lizarralde, F., Hsu, L., and dos Reis, N. R. S. (2010). Kinematic reconfigurability control for an environmental mobile robot operating in the amazon rain forest. *Journal of Field Robotics*, 27(2):197--216.

Freitas, G., Rocha, F. A., Torre, M. P., Fontes Junior, A. F., Ramos, V. R., Nogueira, L. E., Costa, D., Santos, A., Cota, E., Miola, W., Azpurua, H., et al. (2018). Multi-terrain inspection robotic device and methods for configuring and guiding the same. US Patent App. 16/485,397, WIPO Patent No. 2018145183.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209--226.

Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: a survey. *Artificial intelligence review*, 43(1):55--81.

Gallego, G. and Scaramuzza, D. (2017). Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, 2(2):632--639.

Garland, M. (1999). Quadric-based polygonal surface simplification. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

Gonçalves, V. M., Pimenta, L. C. A., Maia, C. A., Dutra, B. C. O., and Pereira, G. A. S. (2010). Vector fields for robot navigation along time-varying curves in n-dimensions. *IEEE Transactions on Robotics*, 26(4):647–659. ISSN 1552-3098.

Government of Ontario, Ministry of Labour, O. H. and Branch, S. (2020). Confined spaces: Confined spaces guideline: Ontario ministry of labour.

Heberger, J. (2018). Demonstrating the financial impact of mining injuries with the “safety pays in mining” web application. *Mining engineering*, 70(12):37.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71--78.

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*. Software available at `http://octomap.github.com`.

Howard, A. and Roy, N. (2003). The Robotics Data Set Repository (Radish).

Hull, B. P., Leigh, J., Driscoll, T. R., and Mandryk, J. (1996). Factors associated with occupational injury severity in the new south wales underground coal mining industry. *Safety Science*, 21(3):191--204.

Ishigami, G., Nagatani, K., and Yoshida, K. (2007). Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2361--2366. IEEE.

Jeddisaravi, K., Alitappeh, R., and Guimarães, F. (2016). Multi-objective mobile robot path planning based on A* search. In *Computer and Knowledge Engineering (ICCKE), 2016 6th International Conference on*, pages 7--12. IEEE.

Juliá, M., Gil, A., and Reinoso, O. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427--444.

Jun, J.-Y., Saut, J.-P., and Benamar, F. (2016). Pose estimation-based path planning for a tracked mobile robot traversing uneven terrains. *Robotics and Autonomous Systems*, 75:325--339.

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

Karaman, S. and Frazzoli, E. (2011b). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846--894.

Karlsson, R. and Gustafsson, F. (2017). The future of automotive localization algorithms: Available, reliable, and scalable localization: Anywhere and anytime. *IEEE signal processing magazine*, 34(2):60--69.

Katz, S., Tal, A., and Basri, R. (2007). Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*, pages 24--es. ACM.

Kazhdan, M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1--13.

Kraft, D. et al. (1988). A software package for sequential quadratic programming. *DFVLR Obersfaffeuhofen.*

Kuipers, B. and Byun, Y.-T. (1988). A robust, qualitative method for robot spatial learning. In *AAAI*, volume 88, pages 774--779.

Labbé, M. and Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *J. of Field Robotics*, 36(2):416--446.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. *Business & information systems engineering*, 6(4):239--242.

LaValle, S. M. et al. (1998). Rapidly-exploring random trees: A new tool for path planning.

Leingartner, M., Maurer, J., Ferrein, A., and Steinbauer, G. (2016). Evaluation of sensors and mapping approaches for disasters in tunnels. *Journal of field robotics*, 33(8):1037--1057.

Li, M.-g., Zhu, H., You, S.-z., Wang, L., Zhang, Z., and Tang, C.-q. (2019). Imu-aided ultra-wideband based localization for coal mine robots. In *International Conference on Intelligent Robotics and Applications*, pages 256--268. Springer.

Liepa, P. (2003). Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 200--205.

Lluvia, I., Lazkano, E., and Ansuategi, A. (2021). Active mapping and robot exploration: A survey. *Sensors*, 21(7):2445.

Lu, F. and Milios, E. (1997). Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic systems*, 18(3):249--275.

Macenski, S., Martin, F., White, R., and Ginés Clavero, J. (2020). The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Maffei, R., Souza, M. P., Mantelli, M., Pittol, D., Kolberg, M., and Jorge, V. A. (2020). Exploration of 3d terrains using potential fields with elevation-based local distortions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4239--4244. IEEE.

Maimone, M. W., Leger, P. C., and Biesiadecki, J. J. (2007). Overview of the mars exploration rovers' autonomous mobility and vision capabilities. In *IEEE international conference on robotics and automation (ICRA) space robotics workshop*.

Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F. (2002). An experiment in integrated exploration. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 1, pages 534--539. IEEE.

Marquez, A., Tank, B., Meghani, S. K., Ahmed, S., and Tepe, K. (2017). Accurate uwb and imu based indoor localization for autonomous robots. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1--4. IEEE.

Martz, J., Al-Sabban, W., and Smith, R. N. (2020). Survey of unmanned subterranean exploration, navigation, and localisation. *IET Cyber-systems and Robotics*, 2(1):1--13.

Meagher, D. (1982). Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129--147.

Messuri, D. and Klein, C. (1985). Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion. *IEEE Journal on Robotics and Automation*, 1(3):132--141.

Miettinen, K. (2012). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

Miller, I. D., Cladera, F., Cowley, A., Shivakumar, S. S., Lee, E. S., Jarin-Lipschitz, L., Bhat, A., Rodrigues, N., Zhou, A., Cohen, A., et al. (2020). Mine tunnel exploration using multiple quadrupedal robots. *IEEE Robotics and Automation Letters*, 5(2):2840--2847.

Ministry of Labour, o. B. (2006). Norma regulamentadora 33, portaria sit n.º 202. Acessado: 26 fev. 2021.

Mitchell, R. J., Driscoll, T., and Harrison, J. E. (1998). Traumatic work-related fatalities involving mining in australia. *Safety science*, 29(2):107--123.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 116--121. IEEE.

Morris, A., Ferguson, D., Omohundro, Z., Bradley, D., Silver, D., Baker, C., Thayer, S., Whittaker, C., and Whittaker, W. (2006). Recent developments in subterranean robotics. *Journal of Field Robotics*, 23(1):35--57.

Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255--1262.

Murphy, R. R. (2014). *Disaster robotics.* MIT press.

Newman, P., Bosse, M., and Leonard, J. (2003). Autonomous feature-based exploration. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 1234--1240. IEEE.

Ohradzansky, M. T., Mills, A. B., Rush, E. R., Riley, D. G., Frew, E. W., and Humbert, J. S. (2020). Reactive control and metric-topological planning for exploration. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4073--4079. IEEE.

Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., and Nieto, J. (2017). Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 Ieee/rsj International Conference on Intelligent Robots and Systems (iros)*, pages 1366--1373. IEEE.

Otte, S., Kulick, J., Toussaint, M., and Brock, O. (2014). Entropy-based strategies for physical exploration of the environment's degrees of freedom. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 615--622. IEEE.

Papachristos, C., Khattak, S., and Alexis, K. (2017). Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 4568--4575. IEEE.

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

Papachristos, C., Mascarich, F., Khattak, S., Dang, T., and Alexis, K. (2019b). Localization uncertainty-aware autonomous exploration and mapping with aerial robots using receding horizon path-planning. *Autonomous Robots*, 43(8):2131--2161.

Paton, M., Strub, M. P., Brown, T., Greene, R. J., Lizewski, J., Patel, V., Gammell, J. D., and Nesnas, I. A. (2020). Navigation on the line: Traversability analysis and path planning for extreme-terrain rappelling rovers. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*. IEEE.

Paz-Delgado, G., Azkarate, M., Sánchez-Ibáñez, J., Pérez-del Pulgar, C., Gerdes, L., and García-Cerezo, A. (2020). Improving autonomous rover guidance in round-trip missions using a dynamic cost map. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7014--7019. IEEE.

Peynot, T., Underwood, J., and Scheding, S. (2009). Towards reliable perception for unmanned ground vehicles in challenging conditions. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1170--1176. IEEE.

Pimentel, J. M., Alvim, M. S., Campos, M. F., and Macharet, D. G. (2018). Information-driven rapidly-exploring random tree for efficient environment exploration. *Journal of Intelligent & Robotic Systems*, 91(2):313--331.

Prágr, M., Čížek, P., Bayer, J., and Faigl, J. (2019). Online incremental learning of the terrain traversal cost in autonomous exploration. In *Robotics: Science and Systems (RSS)*.

Prágr, M., Čížek, P., and Faigl, J. (2018). Cost of transport estimation for legged robot based on terrain features inference from aerial scan. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1745--1750. IEEE.

Pütz, S., Wiemann, T., and Hertzberg, J. (2021). The mesh tools package introducing annotated 3d triangle maps in ros. *Robotics and Autonomous Systems*, page 103688.

Pütz, S., Wiemann, T., Sprickerhof, J., and Hertzberg, J. (2016). 3d navigation mesh generation for path planning in uneven terrain. *IFAC-PapersOnLine*, 49(15):212--217.

Qin, H., Meng, Z., Meng, W., Chen, X., Sun, H., Lin, F., and Ang, M. H. (2019). Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments. *IEEE Transactions on Vehicular Technology*, 68(2):1339--1350.

Raja, R., Dutta, A., and Venkatesh, K. (2015). New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover. *Robotics and Autonomous Systems*, 72:295--306.

Rasshofer, R. H., Spies, M., and Spies, H. (2011). Influences of weather phenomena on automotive laser radar systems. *Advances in Radio Science*, 9(B. 2):49--60.

Rauscher, G., Dube, D., and Zell, A. (2016). A comparison of 3d sensors for wheeled mobile robots. In *Intelligent Autonomous Systems 13*, pages 29--41. Springer.

Regulations, U. C. S. (1997). The confined spaces regulations 1997.

Rezende, A. M., Júnior, G. P., Fernandes, R., Miranda, V. R., Azpúrua, H., Pessin, G., and Freitas, G. M. (2020). Indoor localization and navigation control strategies for a mobile robot designed to inspect confined environments. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1427--1433. IEEE.

Rineau, L. and Yvinec, M. (2007). A generic software design for delaunay refinement meshing. *Computational Geometry*, 38(1-2):100--110.

Rocha, F., Azpúrua, H., Garcia, G., Cota, E., Costa, R. R., and Freitas, G. (2019). Análise e comparaçao de mobilidade de um robô com locomoçao reconfigurável. *2019 Simpósio Brasileiro de Automação Inteligente (SBAI)*.

Roos, F., Bechter, J., Knill, C., Schweizer, B., and Waldschmidt, C. (2019). Radar sensors for autonomous driving: Modulation schemes and interference mitigation. *IEEE Microwave Magazine*, 20(9):58--72.

Ruetz, F., Hernández, E., Pfeiffer, M., Oleynikova, H., Cox, M., Lowe, T., and Borges, P. (2019). Ovpc mesh: 3d free-space representation for local ground vehicle navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8648--8654. IEEE.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1--4. IEEE.

Sack, D. and Burgard, W. (2004). A comparison of methods for line extraction from range data. *IFAC Proceedings Volumes*, 37(8):728--733.

Sanmiquel, L., Bascompta, M., Rossell, J. M., Anticoi, H. F., and Guash, E. (2018). Analysis of occupational accidents in underground and surface mining in spain using data-mining techniques. *International journal of environmental research and public health*, 15(3):462.

Santos, A. S., Azpúrua, H., Pessin, G., and Freitas, G. M. (2019). Planejamento de caminhos para robôs móveis em ambientes acidentados. *SBAI 2019*.

Santos, A. S., Azpúrua, H. I. P., Pessin, G., and Freitas, G. M. (2018). Path planning for mobile robots on rough terrain. In *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, pages 265--270. IEEE.

Saputra, M. R. U., Markham, A., and Trigoni, N. (2018). Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1--36.

Saulnier, K., Atanasov, N., Pappas, G., and Kumar, V. (2020). Information theoretic active exploration in signed distance fields. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Schadler, M., Stückler, J., and Behnke, S. (2014). Rough terrain 3d mapping and navigation using a continuously rotating 2d laser scanner. *KI-Künstliche Intelligenz*, 28(2):93--99.

Scherer, S. A., Dube, D., and Zell, A. (2012). Using depth in visual simultaneous localisation and mapping. In *2012 IEEE International Conference on Robotics and Automation*, pages 5216--5221. IEEE.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, 42(3):1--21.

Schwarz, M. and Behnke, S. (2014). Local navigation in rough terrain using omnidirectional height. In *ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1--6. VDE.

Shan, T. and Englot, B. (2018). Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758--4765. IEEE.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27(3):379--423.

Sharma, S. and Tiwari, R. (2016). A survey on multi robots area exploration techniques and algorithms. In *2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*, pages 151--158. IEEE.

Shen, S., Michael, N., and Kumar, V. (2012). Autonomous indoor 3d exploration with a micro-aerial vehicle. In *2012 IEEE international conference on robotics and automation*, pages 9--15. IEEE.

Silver, D., Bradley, D., and Thayer, S. (2004). Scan matching for flooded subterranean voids. In *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, volume 1, pages 422--427. IEEE.

Singh, S., Simmons, R., Smith, T., Stentz, A., Verma, V., Yahja, A., and Schwehr, K. (2000). Recent progress in local and global traversability for planetary rovers. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1194--1200. IEEE.

Solomon, J. H. and Hartmann, M. J. (2006). Robotic whiskers used to sense features. *Nature*, 443(7111):525--525.

Stentz, A. et al. (1995). The focussed d^* algorithm for real-time replanning. In *IJCAI*, volume 95, pages 1652--1659.

Strub, M. P. and Gammell, J. D. (2020). Advanced bit*(abit*): Sampling-based planning with advanced graph-search techniques. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 130--136. IEEE.

Szrek, J., Trybała, P., Góralczyk, M., Michalak, A., Ziketek, B., and Zimroz, R. (2021). Accuracy evaluation of selected mobile inspection robot localization techniques in a gnss-denied environment. *Sensors*, 21(1):141.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21--71.

Thrun, S. et al. (2002). Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1.

Thrun, S., Thayer, S., Whittaker, W., Baker, C., Burgard, W., Ferguson, D., Hahnel, D., Montemerlo, D., Morris, A., Omohundro, Z., et al. (2004). Autonomous exploration and mapping of abandoned mines. *IEEE Robotics & Automation Magazine*, 11(4):79--91.

Vasquez-Gomez, J. I., Sucar, L. E., and Murrieta-Cid, R. (2013). Hierarchical ray tracing for fast volumetric next-best-view planning. In *2013 International Conference on Computer and Robot Vision*, pages 181--187. IEEE.

Vasquez-Gomez, J. I., Sucar, L. E., Murrieta-Cid, R., and Lopez-Damian, E. (2014). Volumetric next-best-view planning for 3d object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 11(10):159.

Wang, C., Meng, L., Li, T., De Silva, C. W., and Meng, M. Q.-H. (2017). Towards autonomous exploration with information potential field in 3d environments. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 340--345. IEEE.

Wisth, D., Camurri, M., and Fallon, M. (2021). Vilens: Visual, inertial, lidar, and leg odometry for all-terrain legged robots. *arXiv preprint arXiv:2107.07243*.

Witting, C., Fehr, M., Bähnemann, R., Oleynikova, H., and Siegwart, R. (2018). History-aware autonomous exploration in confined environments using mavs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1--9. IEEE.

Wong, U., Morris, A., Lea, C., Lee, J., Whittaker, C., Garney, B., and Whittaker, R. (2011). Comparative evaluation of range sensing technologies for underground void modeling. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3816--3823. IEEE.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence* *in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146--151. IEEE.

Yang, S., Yang, S., and Yi, X. (2018). An efficient spatial representation for path planning of ground robots in 3d environments. *IEEE Access*, 6:41539--41550.

Yi, J., Zhang, J., Song, D., and Jayasuriya, S. (2007). Imu-based localization and slip estimation for skid-steered mobile robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2845--2850. IEEE.

Zhu, C., Ding, R., Lin, M., and Wu, Y. (2015). A 3d frontier-based exploration tool for mavs. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 348--352. IEEE.

Appendix A

The EspeleoRobô robotic platform

Aiming to solve some of the challenges of inspecting confined spaces, Vale S.A and the ITV developed the EspeleoRobô robotic platform [Freitas et al., 2018; Azpurua et al., 2019]. The Espeleorobô is a robotic device specially designed for inspecting confined spaces in mining operations (Figure A.1).

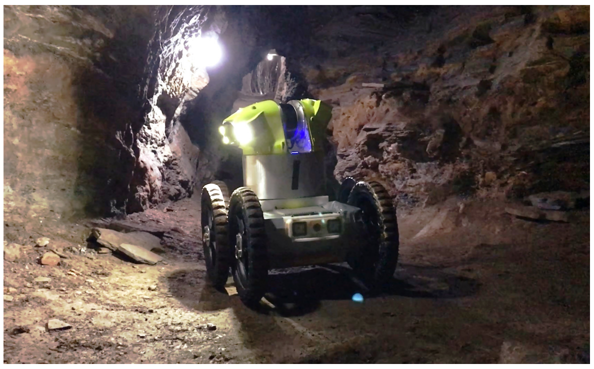


Figure A.1: EspeleoRobô with the modular mapping unit exploring a mining cave.

The robot has six motors with versatile locomotion configurations, as shown in Figure A.2. The fast-swap mechanism allows the robot to have hybrid locomotion systems and a wide variety of configurations to adapt better to the mission's environment directly in the field.

The hardware embedded inside the robot includes multiple cameras, a high-power computer, LiDAR, an IMU, and gas sensors. Given the device's modular construction, it is designed to be easily customizable to adapt to new sensors and mechanic improvements. With the onboard sensorial capabilities, the robot can map the environment in monochromatic and colored 3D point clouds in real-time. The platform is entirely compatible with Robot Operating System (ROS) and runs on open-source software.

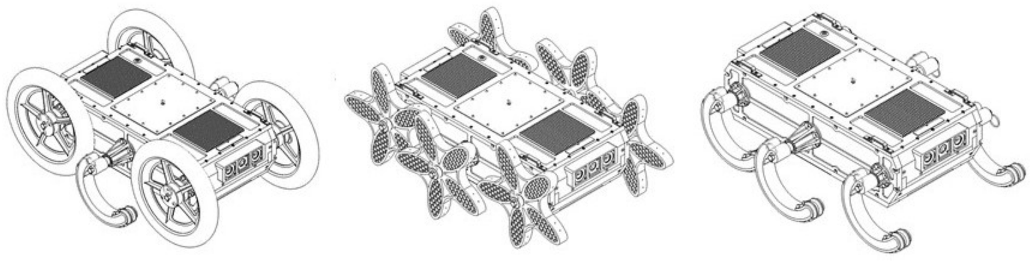


Figure A.2: A sub-set of possible configurations for the robot locomotion systems: circular wheels, star-shaped wheels, legs, and hybrid configurations.

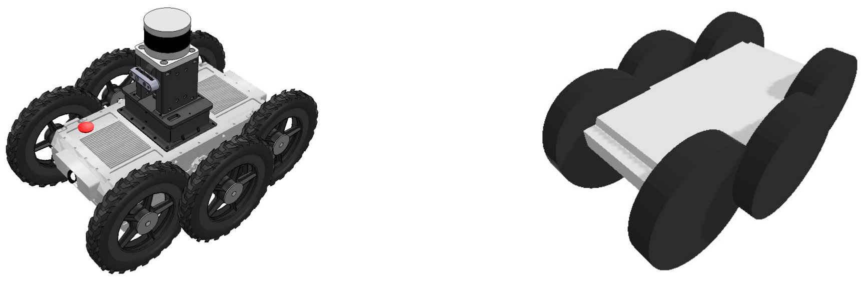
The device was initially used to inspect natural caves during teleoperated missions. However, it is now also used to monitor dam galleries and pipes, among other industrial confined spaces during teleoperated missions. Given the IP67 water and dust protection, the robot can perform in a wide range of environments without the risk of damage. However, since only a single robot has limited reach and exploring capabilities, the research presented in this dissertation uses the Espeleorobô as one of the various robotic platforms for testing and validating the proposed methods for cooperation and collaboration.

Appendix B

Expanded terrain interaction modeling

In this chapter, we extend the description of the terrain interaction models used for estimating the final pose of the robot at a specific location on the map. The following methods require the yaw angle γ of the robot, which is estimated by the direction the robot took while performing the path: we use the previous parent nodes and edges to estimate this direction.

Terrain interaction with a 3D model For this interaction, we need an accurate 3D model of the robot, capable of representing the platform's dynamics correctly but simple enough to not produce extra slowness due to an excess of details. In this sense, we simplified the original Compter Assisted Design (CAD) model of the robot to the geometric primitives that fitted better to improve computations (Figure B.1). Of course, if there are no limitations in CPU resources and time, the complete detailed model could be used instead.



(a) Detailed robot model. (b) Simplified model.

Figure B.1: Different 3D robot model versions: (a) detailed model and (b) simplified model.

The interactions between the robot's 3D model and the terrain are performed by a realistic physics simulator: PyBullet (Bullet3) [Coumans and Bai, 2019]. In this sense, given the n_{goal} 3D position and a yaw angle γ, we deploy the robot from a 1 m height of n_{goal} on the simulator and estimate the Δ of the movements in the z axis (when the robot stopped after ground contact), pitch α, and roll β (when the robot stopped its angular movements). Then that delta stabilizes we finalize the simulation and estimate the z normal vector of the current robot position over the terrain (Figure B.2). To prevent any undesirable movements on the linear x and y axis, the simulator movements were limited to only linear z and angular α, β, γ movements using invisible prismatic joints between the robot and the world at the robot definition document. The complete process is described in Algorithm 8, where $\mathbb{M}$ is the terrain mesh, n_x^i, n_y^i, n_z^i are the actual node centroid location, t_{iter} is the maximum iteration number, τ_z is the threshold for the z axis and τ_{quat} is the threshold for the angle. The *quaternionAboluteDistance* function estimates the the chord of the shortest path/arc that connects two quaternions, thus giving a good indicator of rotation similarities. The "$\cdots$" denote the filepath to the mesh files.

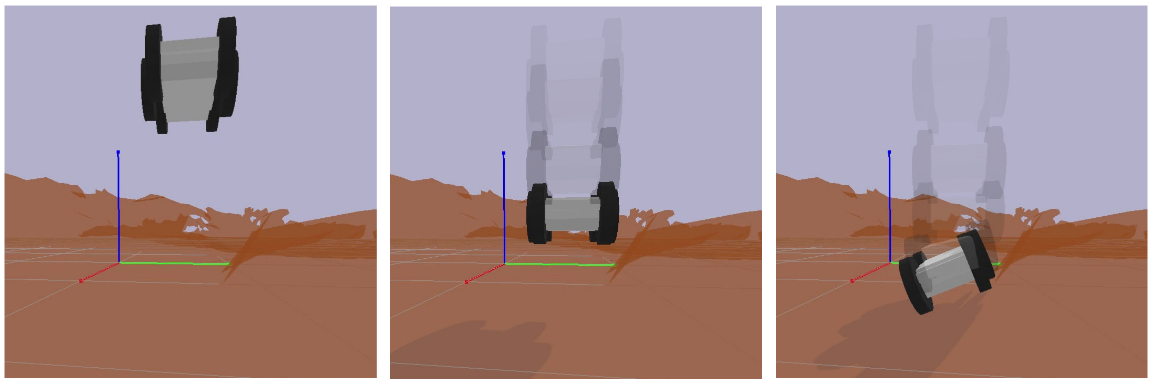

(a) Start position (1 m above ground). (b) Mid air. (c) Final pose after contact with the ground.

Figure B.2: Iterative process of estimating the interaction of a complex 3D model of the robot with the terrain.

The principal handicap of the complete 3D model terrain interaction is processing time. As this method is needed to be executed for all nodes visited in the graph search algorithm, the process will take a considerable amount of time given the graph's size, despite the optimizations made for speeding up the entire process. In this sense, a middle ground between computational complexity and accuracy may be more convenient for most cases.

Algorithm 8: Robot's 3D model terrain interaction estimation

```
$\mathbb{M} \leftarrow loadTerrain(\cdots)$
$R \leftarrow loadRobot(\cdots, n^i_x, n^i_y, n^i_z + 1.0, \gamma_{init})$   ▷ Spawn 1 meter above the node
$\hat{R} \leftarrow R$
for $i \leftarrow 1$ to $t_{iter}$ do
    $R \leftarrow stepSimulation(R)$
    $\Delta z \leftarrow R_z - \hat{R}_z$
    $\Delta quat \leftarrow quaternionAboluteDistance(R_{quat}, \hat{R}_{quat})$
    if $\Delta z \leq \tau_z$ and $\Delta quat \leq \tau_{quat}$ then
        break                                          ▷ Robot has stopped moving
    end
    $\hat{R} \leftarrow \mathbb{R}$
end
return $R_{quat}$                                      ▷ Return the angular pose
```

Terrain interaction with a support polygon This interaction with the environment is a middle ground in computational complexity between the proposed detailed robot's 3D model interaction (CPU intensive) and the single point interaction ($\mathcal{O}(1)$). We propose using the robot's support polygon $\mathcal{SP}$ to estimate the most appropriate set of configurations that accommodate it to the terrain topography via optimization. Since the support polygon uses the robot's physical configuration to represent the contact points with the ground, this method is a reasonable choice to obtain an accurate position without the overhead of using a complex 3D model.

In this sense, the terrain vertexes extracted by a radius of the point of interest (n_i) are modeled as a continuous interpolated function $h(x, y)$ using a piecewise cubic curvature-minimizing interpolant, where x and y are world coordinates. The interpolant uses the Qhull triangulator [Barber et al., 1996] over the small subset of terrain points and calculates a piecewise cubic interpolating Bezier polynomial on each triangle using the Clough-Tocher algorithm [Alfeld, 1984].

The support polygon $\mathcal{SP}$ can be represented in the world $\mathcal{W} \in \mathbb{R}^3$ as $(x, y, z, \alpha, \beta, \gamma)$ where (x, y, z) are the coordinates of the center of the polygon and (α, β, γ) are the roll, pitch, and yaw angles. Since we are interested in finding the best configuration that aligns the polygon to the terrain, we need to find the best (z, α, β) configuration for a given (z, y, γ) [Jun et al., 2016]. In this sense, the objective of the proposed model is to find the configuration that minimizes the distance between the vertices of the $\mathcal{SP}$ and the ground represented by the interpolated function $h(x, y)$:

$$min \sum_{i \in |\mathcal{SP}|} \mathcal{SP}^i_z - h(\mathcal{SP}^i_x, \mathcal{SP}^i_y). \qquad \text{(B.1)}$$

$$
\begin{aligned}
\text{subject to: } \mathcal{SP}_z^i &\geq 0 \ \forall i \in |\mathcal{SP}|, \\
-\pi &\leq \alpha \leq \pi, \\
-\pi &\leq \beta \leq \pi, \\
-2 &\leq Z \leq 2,
\end{aligned}
$$

using the optimization Sequential Least SQuares Programming (SLSQP) Algorithm [Kraft et al., 1988]. The complete process of pose optimization using the $\mathcal{SP}$ can be observed in Figure B.3. This method is considerably faster than the 3D model interaction, but at the cost of accuracy: since only the polygon's border points are considered in the optimization, any object inside convex-hull could extrapolate the height of the polygon and go through it. Nevertheless, in most cases, this seems to be a reasonable tradeoff given the robot's limited computing resources.

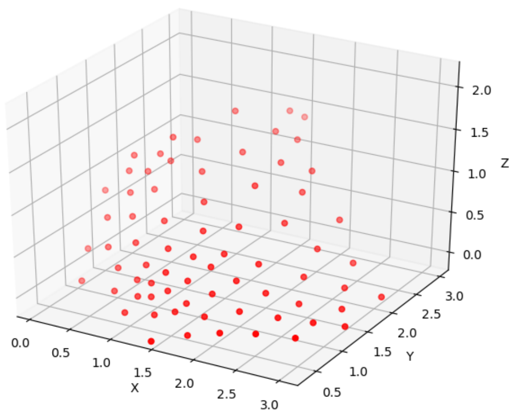
Selecting the adequate terrain interaction model The correct terrain interaction model's correct selection is critical for successful path planning in rugged terrains. In this regard, we propose selecting the metrics based upon the variance of the normal between the direct and second-order neighbors of the node of interest (n_i). To decrease the influence of nodes further away than ≈ 2.0 m from n_i we implemented a decay function (Figure B.4). The Gaussian function with $\mu = 0$ and $\sigma^2 = 0.3$ was chosen as the decay function given the smooth curve, rendering the influence of nodes further than ≈ 1.8 m almost negligible:

$$
\text{GaussianDecay}(x, \mu, \sigma^2) = \frac{\exp((x-\mu)^2)}{2 * \sigma^2}. \tag{B.2}
$$

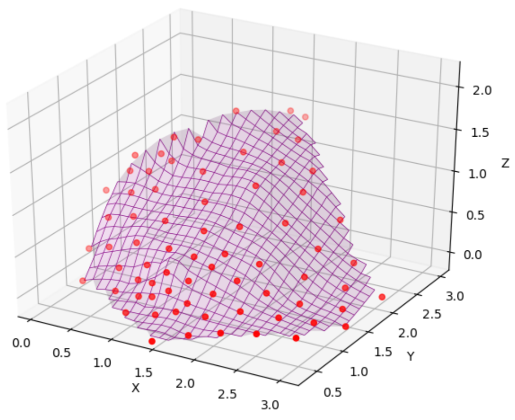
We define a threshold value $\tau_{terrain}$ for selecting the point terrain interaction (the simplest one) when the terrain is predominantly equal in inclination around n_i. The model selection intends to simplify calculations when there are high chances of the terrain being planar so that the more direct point interaction will be sufficient. Then $\tau_{terrain}$ is extrapolated another of the more realistic methods is selected. Finally, the terrain interaction model is selected using the following equation:

$$
\text{terrainIntModel}(n_i) = \begin{cases} \textit{point}, & \text{if neigboursAngleDev}(n_i) \leq \tau_{terrain} \\ \textit{realistic}, & \text{otherwise.} \end{cases} \tag{B.3}
$$

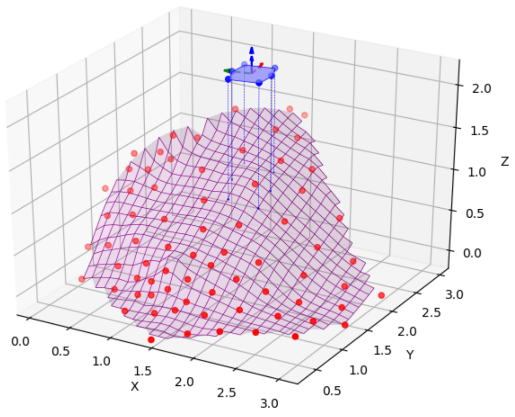
Algorithm 9 shows the process of estimating the angle variation of the neighbors



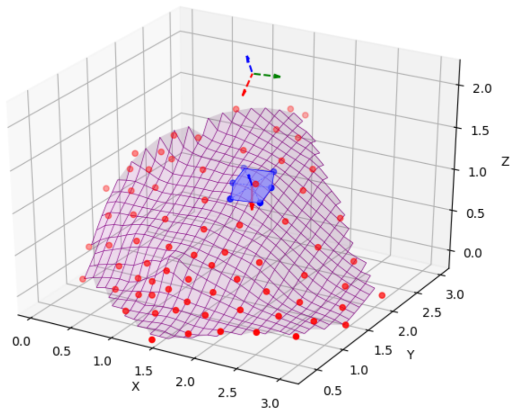
(a) Node vertexes.



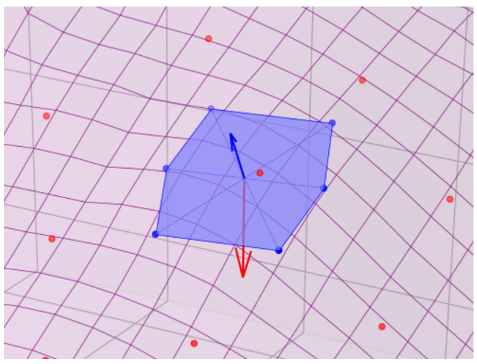
(b) Interpolated terrain.



(c) Support polygon start pose.



(d) Final pose after contact with the ground.



(e) Zoom-in at the final pose.

Figure B.3: Iterative process of estimating the interaction of a the robot's support polygon $\mathcal{SP}$ with the interpolated sub-set of terrain points.

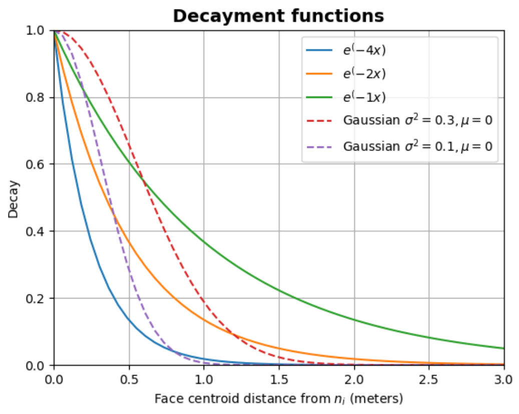


Figure B.4: Decay functions showing the angle variation influence associated with neighboring nodes' euclidean distance from n_i.

using the weighted standard deviation, where $\mathcal{M}$ is the list of second-order neighbors of the node of interest n_i, $\mathcal{Z}$ the list of face's angles and $\mathcal{W}$ the list of face's weights.

Algorithm 9: Neighbors weighted angle deviation estimation

```
$\mathcal{M} \leftarrow getSecondOrderNeighbors(n_i)$
$\mathcal{F} \leftarrow \emptyset$                                        ▷ Face's angles
$\mathcal{W} \leftarrow \emptyset$                                        ▷ Averaging weights
for $j \leftarrow 1$ to $|\mathcal{M}|$ do
    $m \leftarrow \mathcal{M}_j$
    $\mathcal{F}_z \leftarrow \mathcal{F}_z \cup \vec{m}_z$
    $dist \leftarrow \text{euclideanDist}(m, n_i)$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \text{gaussianDecay}(dist)$
end
$\mathcal{F}_{avg} \leftarrow \text{weightedAvg}(\mathbb{F}_z, \mathcal{W})$
$\mathcal{F}_{var} \leftarrow \text{weightedAvg}((\mathbb{F}_z - \mathcal{F}_{avg})^2, \mathcal{W})$
$\mathcal{F}_{std} \leftarrow \sqrt{\mathcal{F}_{var}}$                  ▷ Weighted standard deviation
return $\mathcal{Z}_{std}$
```