Map Quality Criteria for Autonomous Exploration in Natural Environment

Stéphanie Aravecchia

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Map Quality Criteria for Autonomous Exploration in Natural Environment

THÈSE

présentée et soutenue publiquement le 8 Décembre 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention automatique, traitement du signal et des images, génie informatique)

par

Stéphanie Aravecchia

Composition du jury

Présidente :	Ouiddad Labbani-Igbida
Rapporteurs :	Ouiddad Labbani-Igbida Bruno Vallet
Examinateur :	Roland Lenain
Directeur de thèse :	Cédric Pradalier
Co-directrice de thèse :	Marianne Clausel

Laboratoire IRL 2958 GeorgiaTech - CNRS

Georgia Tech

CNRS

International Research Laboratory

Georgia Tech • CNRS IRL 2958

Abstract

Autonomous exploration of an unknown environment consists in answering the question, "where to go next?" Beginning in a completely unknown place, the robot selects its next destinations and navigates towards them, simultaneously discovering its surroundings. This thesis centers on a specific environment type: natural environments like parks or forests. In such environments, what approach should we adopt if our objective is for the robot to conduct its exploration mission with the fundamental goal of creating a map from its observations? Our research main focus lies in establishing a link between the exploration strategy and the resultant map quality. The central challenge stems from the fact that in an unknown environment, by definition, the map's quality cannot be known throughout the exploration task. If a reference map already exists, the task no longer qualifies as an exploration mission.

The major contribution of this thesis lies in associating the map quality with the robot's trajectory in the environment. We establish that locally, the map quality is intricately tied to the robot's viewpoint observations. We introduce a set of viewpoint statistics, each operating as an individual indicator of map quality, which we eventually combine into a predictor. When an area of the map is observed from closer, or from more diverse viewpoints, its quality is enhanced. Ultimately, we integrate this insight into an exploration policy, where the information gain is formulated based on viewpoint statistics. We demonstrate that such an exploration policy indeed produces a better map than traditional baselines.

The second significant contribution of this thesis centers on assessing the map quality itself. Obtaining a meaningful, measure across diverse environments, proved to be a nontrivial task. To address this, we develop a methodology to locally compare the map built from the robot's observations with a reference map. We measure the local map quality with multiple metrics to identify the most robust one.

In addition to the above, this thesis includes the development of an experimental framework, designed for our task. Building upon this, we evaluate our methods both in simulations and real-world experiments.

Keywords: mobile robotics, 3d-mapping, 3d-lidar, next-best-view, active sensing, exploration

Résumé

Explorer de manière autonome un environnement inconnu consiste à répondre à la question "où aller ensuite ?". Partant d'un endroit totalement inconnu, le robot sélectionne ses prochaines destinations et navigue vers elles, découvrant ce-faisant son environnement. Cette thèse se concentre sur un type d'environnement spécifique : les environnements naturels tels que les parcs ou les forêts.

Dans ce type d'environnements, quelle approche devrions-nous adopter si la mission d'exploration du robot a pour objectif principal la création d'une carte, à partir de ses observations ? Notre recherche s'intéresse à l'établissement d'un lien entre la stratégie d'exploration et la qualité de la carte résultante. La principale difficulté est que, dans un environnement inconnu, par définition, la qualité de la carte ne peut pas être connue pendant mission d'exploration. Si une carte de référence existe déjà, la tâche ne peut plus être qualifiée d'exploration.

La principale contribution de ce travail réside dans la démonstration de l'existence d'une relation entre la qualité de la carte et la trajectoire du robot dans l'environnement. Nous établissons que, localement, la qualité de la carte est étroitement liée aux points de vue des observations du robot. Nous introduisons un ensemble de statistiques de points de vue d'observation, chacune fonctionnant comme un indicateur individuel de la qualité de la carte, que nous combinons finalement en un prédicteur. Une zone de la carte observée de plus près, ou à partir de points de vue plus diversifiés, voit sa qualité améliorée. Finalement, nous intégrons cette connaissance dans une politique d'exploration, où le gain d'information est formulé en fonction des statistiques sur les points de vue d'observation. Nous démontrons qu'une telle politique d'exploration produit effectivement une meilleure carte que celles issues de méthodes traditionnelles.

La deuxième contribution significative de cette thèse est relative à l'évaluation de la qualité de la carte elle-même. Obtenir une mesure de qualité significative à travers des environnements divers s'est avéré être une tâche non triviale. Pour y remédier, nous développons une méthodologie pour comparer localement la carte construite à partir des observations du robot avec une carte de référence. Nous mesurons la qualité locale de la carte avec plusieurs métriques pour identifier la plus robuste.

Pour obtenir ces résultats, cette thèse comprend le développement d'un cadre expérimental, conçu pour notre tâche. Sur cette base, nous évaluons nos méthodes à la fois en simulation et lors d'expériences réelles.

Mots-clés: robotique mobile, cartographie 3D, laser 3D, vue suivante optimale, perception active, exploration.

Motivation

Explorer un environnement avec un robot implique de partir d'un état initial où les environs du robot sont totalement inconnus. À mesure que le robot interagit avec l'environnement, il construit progressivement une représentation de ses alentours. Bien que cela puisse sembler simple, plusieurs concepts essentiels doivent être définis.

Tout d'abord, qu'est-ce qui constitue une représentation pertinente des environs du robot ? Est-elle conçue pour que le robot interagisse directement, ou est-elle destinée à être analysée par un opérateur humain ? Répondre à cette question est fondamentalement une question de choix de conception. Il n'y a pas de représentation cartographique universellement meilleure, ce choix dépend des exigences spécifiques.

Deuxièmement, dans un scénario où pratiquement tout autour du robot est inconnu, comment doit-il décider où aller ensuite pour recueillir des informations ? Doit-il se déplacer continuellement, traitant les informations au fur et à mesure de leur arrivée ? Alternativement, doit-il sélectionner des objectifs spécifiques puis traiter les informations en naviguant vers eux ? Cette décision, appelée politique d'exploration, est un autre choix de conception clé.

Dans ce travail, nous abordons ces deux questions dans un contexte unifié. Nous considérons le robot comme un robot terrestre, un véhicule terrestre sans pilote (UGV), et l'environnement inconnu qu'il explore est un environnement naturel, tel qu'un parc ou une forêt, comme illustré par la Figure 1.1. Notre représentation cartographique choisie est une figure familière dans le domaine de la robotique : la grille 3D, discrétisant l'ensemble du volume en voxels. Tout au long de notre travail, nous entreprenons diverses tâches et développements, tous dans le but global de permettre une politique d'exploration qui maximise la qualité de la carte pendant le processus d'exploration.

Ce résumé vise à présenter les divers défis inhérents à cet objectif, qui, à leur tour, motivent la recherche menée dans cette thèse.

Qualité de la carte

Le premier défi réside dans l'évaluation de la qualité de la carte 3D dans des environnements naturels. Généralement, l'évaluation de la qualité de la carte vise à fournir une métrique unique qui reflète la qualité globale de la carte. Cependant, dans le contexte d'un robot cartographiant de manière autonome un environnement naturel à des fins d'inspection ou de surveillance, il devient intéressant d'obtenir une mesure localisée de la qualité de la carte. En effet, la qualité de la carte peut ne pas être nécessairement homogène dans un environnement potentiellement étendu.

Cette recherche se concentre spécifiquement sur ce scénario, où les observations du Lidar 3D du robot construisent la carte. Dans ce cas, la représentation cartographique prédominante est la grille 3D, où chaque voxel encode de l'information. Traditionnellement, cette grille 3D encode la probabilité d'occupation pour chaque voxel. Cependant, dans ce scénario courant, les mesures conventionnelles de qualité de la carte, à savoir la couverture de surface et la précision de la reconstruction, peuvent ne pas toujours avoir un sens significatif, surtout lorsqu'il s'agit d'environnements naturels qui sont non seulement clairsemés, mais aussi non structurés. Nous le démontrerons dans cette thèse, en mettant l'accent sur le cas spécifique de la cartographie d'un environnement clairsemé et non structuré, tel qu'un environnement naturel, en simulation et avec une expérience dans le monde réel.

Dans le cas d'environnements "non structurés", des défis distincts se posent par rapport à ceux rencontrés lors de la cartographie d'environnements "structurés". La littérature se concentre souvent sur la cartographie 3D dans des environnements structurés tels que les zones urbaines. Cependant, en ce qui concerne le travail dans des environnements naturels, la différenciation entre structuré et non structuré, dense et clairsemé, devient plus marquée en raison des défis uniques impliqués. La carte 3D dans un environnement naturel est à la fois non structurée et clairsemée. Elle se compose principalement d'espace vide, avec seulement quelques points où le Lidar 3D touche réellement un objet, compliqué davantage par le niveau de bruit accru inhérent aux environnements naturels. De plus, la localisation du robot dans la carte est également une source d'erreurs, car la carte 3D est une accumulation probabiliste de tous les nuages de points transformés dans le repère dérivé de la localisation. Cette recherche n'évalue pas directement divers algorithmes de cartographie appliqués aux environnements naturels. Au lieu de cela, son accent réside dans la résolution des défis liés à l'évaluation de la qualité des cartes 3D dans de tels environnements. Dans ce travail, nous proposons une méthodologie pour évaluer la qualité de la carte au niveau local. Ensuite, nous évaluons et comparons plusieurs métriques, et proposons de sélectionner une métrique particulière pour sa robustesse dans ce type d'environnement.

Établir un à priori sur la qualité de la carte pour guider l'exploration

Le deuxième défi est plus directement lié au processus d'exploration. L'objectif principal de l'exploration autonome est de répondre à la question "où aller ensuite ?" Lorsqu'un robot explore un espace inconnu, le volume complet est inconnu au début de l'exploration. À mesure que le robot se déplace dans l'espace, il recueille des informations et met à jour une carte.

En robotique, l'exploration autonome consiste à trouver la zone la plus intéressante à visiter, déterminant essentiellement une destination cible. Lorsque l'objectif de l'exploration

est de construire une carte précise, une question centrale se pose : “dans les régions explorées, quelle est la qualité actuelle de la carte ?” De plus, lorsqu'on travaille dans des environnements naturels, les données de référence sont souvent indisponibles. Cela soulève une deuxième question : “comment pouvons-nous dériver une politique d'exploration pour améliorer la qualité de la carte sans accéder à une carte de référence ?”

Une solution pour aborder simultanément ces deux questions est de prédire la qualité de la carte au niveau local à partir des observations du robot. À partir de cette prédiction, nous pouvons dériver une politique d'exploration. C'est le défi que nous avons l'intention de relever dans cette thèse.

Pour relever ce défi, notre travail propose plusieurs statistiques sur les points de vue des observations, efficaces en termes de temps de calcul, qui offrent des informations sur la qualité de la carte au niveau local. Ces statistiques mettent en évidence les zones qui méritent d'être revisitées et comment le faire. Dans le même temps, elles aident à identifier les zones qui n'améliorent pas significativement la qualité de la carte, permettant de les écarter. Notre travail démontre statistiquement que ces statistiques sur points de vue d'observation présument de la qualité locale de la carte.

De plus, nous intégrons ces statistiques sur les points de vue d'observation dans une politique d'exploration. Une politique d'exploration est un équilibre entre un coût, généralement le coût pour atteindre un objectif, et un gain attendu, qui représente la connaissance attendue que le robot acquerra en atteignant l'objectif. Le meilleur objectif selon la politique est le prochain meilleur point de vue (Next-Best-View). Dans ce travail, nous incorporons ces statistiques dans la sélection du prochain meilleur point de vue. Ce faisant, nous démontrons que lorsque la politique d'exploration formule son gain d'information en fonction des statistiques sur les points de vue d'observation, la qualité globale de la carte s'améliore.

Plan de la thèse

Chapitre 2 introduit la théorie de fond nécessaire à la compréhension de la recherche présentée dans cette thèse. Il commence par donner un aperçu des robots mobiles, présentant leurs capteurs et les défis liés à la localisation. Le chapitre introduit ensuite la cartographie, mettant en évidence les défis inhérents à la tâche, en particulier dans des environnements naturels à la fois clairsemés et non structurés. Il présente également diverses représentations de cartes utilisées en robotique. Enfin, ce chapitre introduit des concepts essentiels liés à l'exploration autonome, commençant par la navigation autonome et se terminant par l'exploration elle-même, y compris le concept de Next-Best-View.

Chapitre 3 pose les bases pour le reste de notre recherche. Lorsque nous avons com-

mencé le travail, nous avons rapidement réalisé l'importance d'une approche locale, tant dans la stratégie d'exploration que dans l'évaluation de la qualité. Pour ce faire, nous divisons l'environnement en de plus petites régions cuboïdes, chacune contenant des informations locales clés. Cette approche localisée forme la base de notre méthodologie. Dans ce chapitre, nous détaillons comment nous extrayons des régions cuboïdes intersectantes de l'environnement réel ou simulé et des interactions du robot avec celui-ci. De plus, nous introduisons dans ce chapitre notre cadre expérimental, avec à la fois des simulations et des expériences dans le monde réel.

Chapitre 4 vise à relever le défi de l'évaluation de la qualité de la carte 3D dans les environnements naturels. Les mesures traditionnelles de qualité de la carte peuvent ne pas être adaptées en raison des caractéristiques spécifiques de ces environnements, telles que le fait qu'ils soient clairsemés et leur absence de structure. Nous évaluons six métriques, conventionnelles et moins conventionnelles, et évaluons leur efficacité dans la mesure de la qualité de la carte. Non seulement nous les évaluons empiriquement en utilisant des régions cuboïdes extraites de cartes de référence et reconstruites, mais nous les évaluons également avec une méthodologie spécifique, impliquant une dégradation itérative de la carte de référence.

Chapitre 5 introduit une approche novatrice pour estimer a priori la qualité de la carte à l'aide de statistiques sur les points de vue d'observation, c'est-à-dire dérivées des données du robot. Cette estimation est réalisée localement sur des régions cuboïdes. Quatre statistiques sont introduites : le nombre d'observations, la distance minimale des observations, le nombre de secteurs angulaires couverts par une observation et la variance sphérique des observations. Nous démontrons statistiquement que ces statistiques sont des indicateurs de la qualité locale de la carte. Enfin, nous entrainons un modèle pour prédire la qualité de reconstruction du cuboïde basée sur ces statistiques, et nous analysons leur importance individuelle. Ce travail jette les bases des stratégies d'exploration, discutées dans le chapitre 6.

Chapitre 6 s'appuie sur les bases établies dans les chapitres précédents pour améliorer la qualité de la carte lors de l'exploration dans des environnements naturels. Nous intégrons les statistiques sur les points de vue d'observation introduites dans le chapitre 5 dans le processus de sélection du Next-Best-View. Ces politiques d'exploration conduisent à une amélioration de la qualité de la carte pendant le processus d'exploration, en particulier dans le type d'environnement considéré dans cette thèse.

Acknowledgments

Avant toute chose, je remercie Nicolas, pour m'avoir soutenue il y a quelques années quand j'ai dit que je voulais quitter mon ancien travail (bien payé) pour aller faire de la robotique (...). Ensuite, pour m'avoir encore soutenue quand j'ai dit que la recherche en robotique, c'était vraiment ce qui me plaisait, mais que bon, pour faire de l'enseignment et de la recherche, ben il fallait un doctorat... Et une thèse, c'est encore trois ans, pas toujours facile (oui, surtout la fin, desolée). Ensuite, je remercie mes enfants. Il n'y a rien de mieux pour relativiser dans la vie que de passer du temps avec ses enfants. Je les remercie pour avoir supporté leur maman pas très très disponible ces derniers temps.

Now, I will switch to English to firstly thank Cédric. Firstly, for giving me the chance to discover robotics and hiring me when I didn't know much, yet still trusting me because I proved I could learn. Then, for showing me how exciting robotics research is, as well as teaching robotics to students. And finally, for giving me the opportunity to pursue this PhD and for funding me, which is not a little thing considering how poorly funded research can be. Then, I would like to thank Marianne, for providing a different perspective on my research. I would not have bet when I started that I would actually enjoy statistics.

Next, I would like to thank all the colleagues who inspired me to pursue a PhD. Assia, for encouraging me and showing me what perseverance means. Antoine, also for encouraging me and for showing how fun field robots are (and, of course, how stubborn).

Lastly, I would like to thank all the people from the lab for the fruitful discussions, and for contributing to the great atmosphere: Pete, Luis, Salim, Othmane, Laura and everyone else.

Pour Arsène, Juliette, Nicolas.

Contents

1

Introduction

1.1 Motivation

Exploring an environment with a robot involves starting from an initial state where the surroundings of the robot are entirely unknown. As the robot interacts with the environment, it gradually builds a representation of its surroundings. While this may seem straightforward, several essential concepts need to be defined.

Firstly, what constitutes a relevant representation of the robot's surroundings? Is it designed for the robot to interact with directly, or is it intended for analysis by a human operator? Answering this question is fundamentally a matter of design choice. There is no universally "best" map representation; it depends on specific requirements.

Secondly, in a scenario where virtually everything around the robot is unknown, how should it decide where to go next to gather information? Should it move continuously, processing information as it arrives? Alternatively, should it select specific goals and then process information as it navigates toward them? This decision, known as the exploration policy, is another key design choice.

In this work, we address both of these questions within a unified context. We consider the robot to be a ground robot, an Unmanned Ground Vehicle (UGV), and the unknown environment it explores is a natural environment, such as a park or a forest, as illustrated with Figure 1.1. Our chosen map representation is a familiar one in robotics: the 3D-grid, discretizing the entire volume into voxels. In the course of our work, we undertake various tasks and developments, all with the overarching aim of enabling an exploration policy that maximizes map quality during the exploration process. This section intends to introduce the various challenges inherent to this objective, which, in turn, motivate the research conducted in this thesis.

Figure 1.1: The Husky Robot with its sensor suite and the type of environment we consider.

Map quality

The first challenge lies in evaluating the 3D-map quality in natural environments. Typically, map quality evaluation aims to provide a single metric that reflects the overall map quality. Nonetheless, in the context of a robot autonomously mapping a natural environment for inspection or monitoring purposes, it becomes interesting to obtain a localized measure of map quality. Indeed, the map quality may not necessarily be homogeneous throughout a possibly large-scale environment.

This research specifically focuses on this scenario, where the robot's 3D-lidar observations construct the map. In this case, the prevailing map representation is the 3D grid, where each voxel encodes information. Traditionally, this 3D grid encodes the occupancy likelihood for each voxel. However, in this common scenario, the conventional measures of map quality, namely surface coverage and reconstruction accuracy, may not always hold significant meaning, especially when dealing with natural environments that are not only sparse but also unstructured. We will demonstrate it in this thesis, by emphasizing the specific case of mapping a sparse, unstructured environment, such as a natural environment, in simulation and with a real world experiment.

In the case of “unstructured" environments, distinct challenges arise compared to those encountered when mapping “structured" environments. The literature often focuses on 3D-mapping in “structured" environments like urban areas. However, when it comes to work in natural environments, the differentiation between structured and unstructured, dense and sparse environments becomes more prominent due to the unique challenges involved. The 3D-map in a natural environment is both unstructured and sparse. It consists predominantly of empty space, with only a few points where the 3D-lidar actually hits an object, further complicated by the increased noise level inherent to natural environments. Additionally, the localization of the robot in the map is also a source of errors because the 3D-map is a probabilistic accumulation of all the point-clouds transformed in the localization frame. This research does not directly evaluate various mapping algorithms applied to natural environments. Instead, its emphasis lies in addressing the challenges associated with assessing the quality of 3D maps in such environments. In this work, we propose a methodology to evaluate the map quality at a local level. Then, we evaluate and compare several metrics, and propose to select one particular metric for its robustness in that type of environment.

Building a prior on map quality to guide the exploration

The second challenge is more directly linked to the exploration process. The primary objective of autonomous exploration is to answer the question "where to go next?" When a robot

explores an unknown space, the complete volume is unknown at the beginning of the exploration. As the robot moves through the space, it gathers information and updates a map.

In robotics, autonomous exploration involves finding the next most interesting area to visit, essentially determining a goal destination. When the goal of exploration is to construct an accurate map, a central question arises: "in regions that have been explored, what is the current map quality?" Moreover, when working in natural environments, ground-truth data is often unavailable. This raises a second question: "how can we derive an exploration policy to improve the map quality without access to any reference map?"

One solution to address both questions simultaneously is to predict the map quality at a local level from the robot's observations. From this prediction, we can derive an exploration policy. This is the challenge we intend to address in this thesis.

To tackle this challenge, our work proposes several computationally efficient viewpoint statistics that offer insights into local map quality. These statistics highlight which areas are worth revisiting and how to do so. Conversely, they help identify areas that do not significantly improve map quality, allowing to discard them. Our work demonstrates statistically that these viewpoint statistics are presumptive of local map quality.

Furthermore, we integrate these viewpoint statistics into an exploration policy. An exploration policy is a balance between a cost, generally the cost of reaching a goal, and an expected gain, which represents the expected knowledge the robot will acquire upon reaching the goal. The best goal according to the policy is the Next-Best-View. In this work, we incorporate these statistics into the selection of the Next-Best-View. By doing so, we demonstrate that when the exploration policy formulates its information gain based on the viewpoint statistics, the overall map quality improves.

1.2 Outline

This section summarizes the thesis, whereas Figure 1.2 shows the interaction between the chapters.

Chapter 2 introduces the background theory required to understand the research presented in this thesis. It begins with an overview of mobile robots, introducing their sensors and the challenges related to localization. The chapter then introduces mapping, highlighting the challenges inherent to the task, particularly in natural environments that are both sparse and unstructured. It also presents various map representations used in robotics. Finally, the chapter introduces essential concepts related to autonomous exploration, beginning with autonomous navigation and ending with exploration itself, including the concept of Next-Best-View.

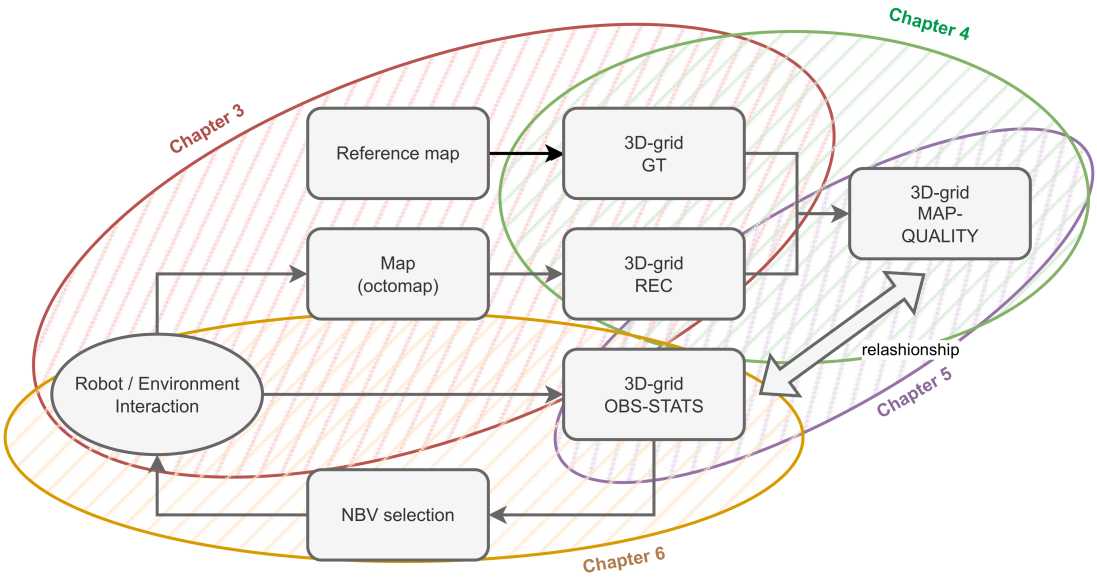

Figure 1.2: System diagram showing the interaction between the different chapters of this thesis.

Chapter 3 lays the groundwork for the rest of our research. When we started the work, we quickly realized the importance of a local approach, both in exploration strategy and quality assessment. To achieve this, we divide the environment into smaller cuboid regions, each containing key local information. This localized approach forms the basis of our methodology. In this chapter, we detail how we extract intersecting cuboid regions from the real or simulated environment and the robot's interactions with it. Moreover, we introduce in this chapter our experimental framework, with both simulations and real-world experiments.

Chapter 4 aims to tackle the challenge of evaluating 3D-map quality in natural environments. Traditional map quality metrics may not be suitable due to the specific characteristics of such environments, like sparsity and lack of structure. We evaluate six metrics, conventional and less conventional, and assess their effectiveness in measuring map quality. Not only we evaluate them empirically using cuboid regions extracted from reference and reconstructed maps, but we also evaluate them with a specific methodology, involving iterative degradation of the reference map.

Chapter 5 introduces an innovative approach to estimate map quality a priori using viewpoint statistics derived from the robot's data. This estimation is made locally on cuboid regions. Four viewpoint statistics are introduced: number of observation, minimum range, number of angular sectors, and spherical variance. We demonstrate statistically that these

statistics are indicators of local map quality. Finally, we train a model to predict cuboid quality based on these statistics, and we analyze their individual importance. This work lays the foundation for the exploration strategies, discussed in Chapter 6.

Chapter 6 builds on the foundation established in the previous chapters to enhance map quality during exploration in natural environments. We integrate the viewpoint statistics introduced in Chapter 5 into the Next-Best-View selection process. These exploration policies result in improved map quality during the exploration process, particularly in the type of environment considered in this thesis.

1.3 Contributions

List of publications

This thesis builds on the following articles:

[4] Aravecchia Stéphanie, Richard Antoine, Clausel Marianne, Pradalier Cédric. "Next-Best-View selection with view-points statistics on the observations". *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Detroit, 2023.

[5] Aravecchia Stéphanie, Richard Antoine, Clausel Marianne, Pradalier Cédric. "Measuring 3D-reconstruction quality in probabilistic volumetric maps with the Wasserstein Distance". *56th International Symposium on Robotics (ISR Europe)*, Stuttgart, 2023.

[3] Aravecchia Stéphanie, Clausel Marianne, Pradalier Cédric. "Comparing Metrics for Evaluating 3D Map Quality in Natural Environment". *Under-review*, 2023.

The following articles are the results of collaborations. Although not directly related to this thesis, they helped me build an understanding of robotics, particularly field robotics, and machine learning challenges.

[60] Richard Antoine, Aravecchia Stéphanie, Geist Mathieu, Pradalier Cédric. "Learning behaviors through physics-driven latent imagination". *Conference on Robot Learning (CORL)*, London, 2021.

[61] Richard Antoine, Aravecchia Stéphanie, Schillaci Thomas, Geist Mathieu, Pradalier Cédric. "How to train your heron". *IEEE Robotics and Automation Letters*, 2021.

[55] Oliveira, Cláudia, Stéphanie Aravecchia, Cédric Pradalier, Vincent Robin, and Simon Devin "The use of remote sensing tools for accurate charcoal kilns' inventory and distribution analysis: Comparative assessment and prospective". *International Journal of Applied Earth Observation and Geoinformation*, 2021.

[49] Mahé Antoine, Richard Antoine, Aravecchia Stéphanie, Geist Mathieu, Pradalier Cédric. "Evaluation of prioritized deep system identification on a path following task", *Journal of Intelligent & Robotic Systems*, 2021

List of open-source repositories

- `https://github.com/stephanie-aravecchia/3d-reconstruction-metrics.git` (chapter 4)
- `https://github.com/stephanie-aravecchia/obs-stats-NBV.git` (chapters 5 and 6)
- `https://github.com/stephanie-aravecchia/octomap_mapping.git` (chapter 3)
- `https://github.com/stephanie-aravecchia/unstructured-env-simulator.git` (chapter 3)
- `https://github.com/stephanie-aravecchia/husky-custom.git` (chapter 3)

2

Principles

2.1 Mobile Robots

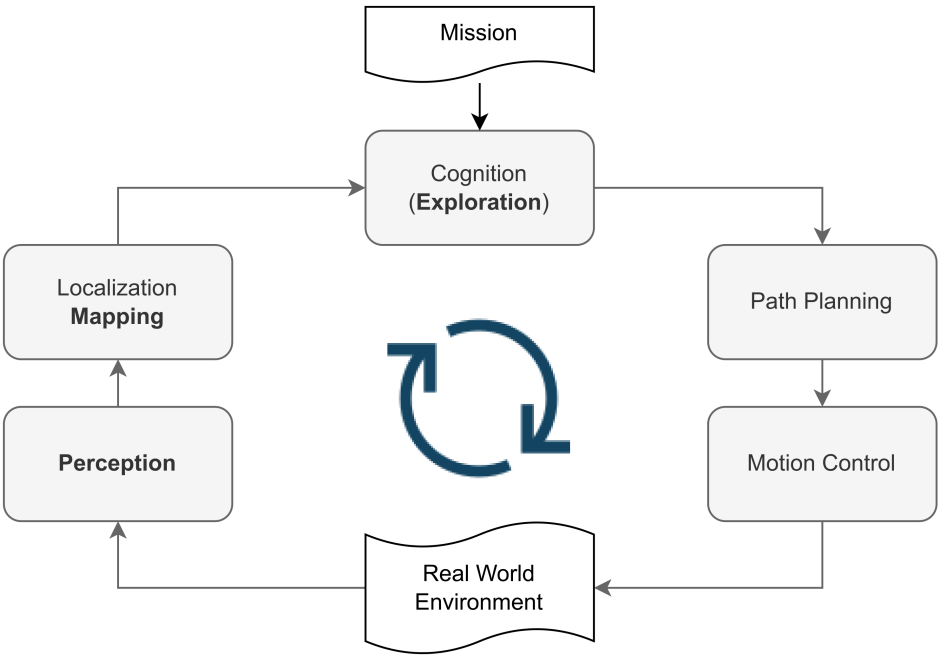

Figure 2.1: Key concepts of mobile robots (adapted from [67]). The concepts used in this work are highlighted in bold in the diagram.

This section is designed to provide an overview of fundamental concepts in mobile robotics that are essential for comprehending the content of this manuscript. The reader familiar with mobile robotics may skip this section.

Typically, in the domain of mobile robots, we categorize their abilities into low-level and high-level functions. This distinction is shown in Figure 2.1, adapted from [67]. This figure illustrates the core concepts of mobile robots. Read from bottom to top, the diagram outlines low-level to high-level abilities.

The lower-level abilities are perception and motion control. They capture the essence of the robot's interaction with the real world. On the perception side, the robot acquires information from its surroundings, whereas on the control side, the robot actions translate into real-world consequences. For instance, by rotating its wheels, the robot moves within its environment. Advancing up the hierarchy in the diagram, we encounter higher-level abilities, such as localization, map building and path planning. Localization, directly linked to perception, represents the robot's ability to determine its own position. Directly connected to perception and localization is map building, a key feature of this work that we will explore in detail later. Similarly, path planning is a high level ability, typically associated to motion control in the context of mobile robots. It represents the robot's ability to navigate robustly

through its surroundings. Reaching the pinnacle of the diagram, cognition tasks come into play. These tasks involve more abstract functions, such as exploration, a core concept of this work we will delve into later. All these concepts find comprehensive explanation in [67].

This work specifically addresses the concepts highlighted in bold within the diagram. These concepts are applied to a wheeled robot, yet the outcomes drawn from our research remain independent of this particular design choice. It is important to note that the field of mobile robots is vast, encompassing various types such as aerial robots, underwater robots, and more. Focusing on ground robots, specifically Unmanned Ground Vehicles (UGVs), there still exists a wide spectrum of locomotion mechanisms, ranging from legged robots to tracked robots, including wheeled robots. The choice of locomotion impacts the right side of the diagram in Figure 2.1. Motion control depends directly on the robot locomotion. Indeed, control laws, commands send to the robot's actuators, heavily depend on the robot locomotion. Even tasks at a higher level, such as path planning, rely on this specific design aspect. For instance, planning a path involving stairs may be achievable for a legged robot, but unrealistic for a one with wheels. This research focuses on the left side of the diagram and on the cognition task on the top. Consequently, while this work is centered on wheeled robots, the conclusions we draw do not solely depend on this specific robot locomotion.

In what follows, we will outline the fundamental concepts introduced here.

2.1.1 Perception

As introduced before, robot perception is a core concept in mobile robots. Every mobile robot is equipped with sensors, and robot sensing is the essence of robot-environment interaction. By taking measurements using various sensors, the robot acquires knowledge about its environment. Often, the sensors are classified between proprioceptive and exteroceptive sensors. While proprioceptive sensors measure values internally to the robot, exteroceptive sensors measure information from the robot's environment. A thorough review of robot sensors is provided in [67]. In what follows, we simply present the common sensors for UGVs, and specifically highlight those necessary to understand this research.

Proprioceptive sensors

Wheeled robots are generally equipped with at least two proprioceptive sensors: one or more wheel encoders and one Inertial Measurement Unit (IMU). A wheel encoder measures the position or the speed of the wheel. In robotics, it is often an optical incremental encoder. It is a device that contains a mechanical light chopper that produces a certain number of pulses for each shaft revolution. By counting the number of pulses, and integrating it in time, the

sensor provides an estimate of the position of the robot. An IMU is a device that contains different sensors: a gyroscope, an accelerometer, and often a magnetometer. They provide a measure of the robot's orientation and inclination. While the measurements are integrated in time, the sensor provides an estimate of the robot position and orientation.

Exteroceptive sensors

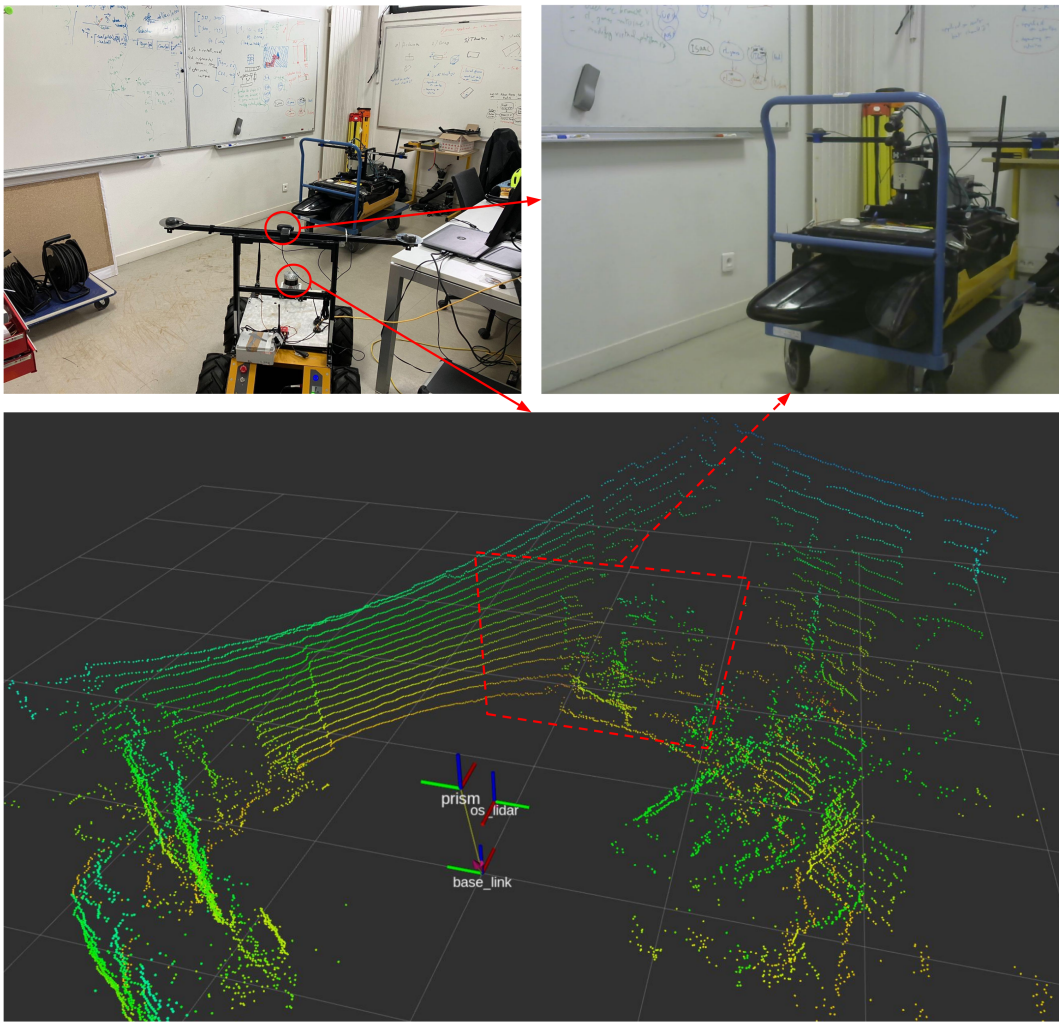

Figure 2.2: Top-left: external view of the scene. Top-right: image from the camera. Bottom: point-cloud from the Lidar. Top-right and bottom correspond to what the robot "sees". The dashed trapezoid provides an idea of the correspondence between camera and point-cloud. The color in the point-cloud simply encodes the height, for visualization purpose.

Exteroceptive sensors are generally split in two categories: passive and active sensors. While passive sensors measure energy coming from the environment, active sensors emit their own energy and measure the response.

In the outdoor application we consider in this work, a common passive exteroceptive sensor used for localization is a GPS (Global Positioning System). The GPS is a constellation of satellites that continuously transmit their location around Earth, in a synchronized way. The GPS receiver reads the transmission from all the visible satellites, and computes its distance to each satellite through the arrival time difference in the signals. By combining information on distance and location of several satellites, it can infer its own position, commonly within a few meters accuracy. To improve the accuracy of the system, the GPS can be improved to RTK-GPS (RTK, Real Time Kinematics). An RTK-GPS system is composed of two receivers: a fixed one, and a mobile one. While the fixed receiver does not move, it computes corrections on the signals it receives from the satellites' constellation. It then sends these corrections to the mobile receiver, that integrates them when inferring its own position. Such a system typically reaches a centimeter accuracy.

Apart from GPS, mobile robots are typically equipped with other exteroceptive sensors directly linked to their task. The most common ones are likely cameras and Lidars. Although common, these two sensors are fundamentally distinct. Firstly, cameras are passive sensors, while Lidars are active sensors. Additionally, their data outputs diverge significantly.

Traditional cameras are sensors that record the light that is reflected on their environment and output an image. This image is a 2D-projection of the environment: a grid of pixels. In traditional cameras, each pixel corresponds to an individual photodetector, that measures the reflected light. This measure is then encoded into a color model, typically RGB (Red Green Blue). A displayed image is the addition of the light beams of those three fundamental colors. Most computer vision applications use RGB images, and RGB cameras are very common sensors in robotics application. One of their main drawback is that they do not provide any information on the distance to the object. All the robot's surroundings are simply projected on a plane. The functioning of camera sensors and their output is thoroughly detailed in [24].

Lidars, for LIght Detection And Ranging, are drastically different. Firstly, as they are active sensors, Lidars generate energy, in this case light, to get a measurement. A Lidar is actually a system emitting light from a rapidly firing rotating laser. This light travels, and eventually hits an object. The reflected light energy then returns to the sensor where it is measured. The sensor measures two quantities. The first is the time it takes for the emitted light to come back: the time of flight. This time is used to calculate the distance, or range. The second is the waveform of the pulse of light that is returned to the sensor. From this waveform, the sensor calculates the intensity of the returned energy. Secondly, the output of a Lidar is fundamentally different from the output of a camera. A 3D-Lidar emits laser rays in known directions. From the direction and the range, the sensor calculates the XYZ coordinates of the returned point expressed in the center of the Lidar frame, along with an intensity value.

From the returned points, the Lidar outputs a point-cloud: an unordered collection of points. Each point in the point-cloud contains XYZ coordinates and the intensity value. Figure 2.3 provides an example, illustrating that only returned points are added to the point-cloud. A description of the technology and its associated challenges can be found in [18], whereas a thoroughly detailed study of Lidars and point-clouds is provided by [76]. This research deals with mapping with a 3D-Lidar.

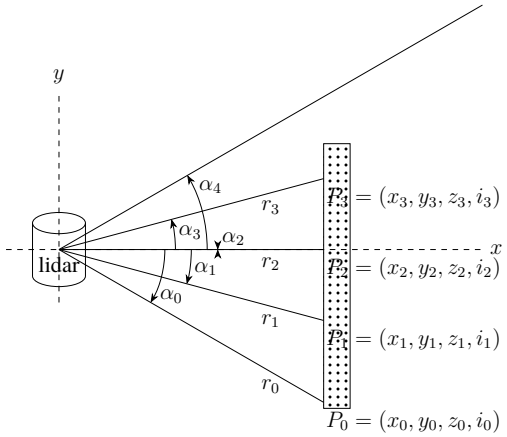

Figure 2.3: This figure illustrates 3D-Lidar. The light rays are emitted from the Lidar, in known directions (angles α). When the ray hits an obstacle, the distance of the returned point is computed (distances r), along with the intensity of the signal i. A point P (x, y, z, i) is added to the point-cloud. Rays that does not return are not in the point-cloud. In this example, although five rays are emitted, the point-cloud contains only four points.

Finally, we would like to emphasize how the data from a camera and a Lidar are different. We do so in Figure 2.2. It shows an external view of the scene, showing the Husky and its sensors, along with the image from the camera, and the point-cloud from the Lidar. The image and the point-cloud are what the robot "sees" with these sensors.

2.1.2 Localization

Another core ability of mobile robots, as introduced earlier, lies in localization. Any image or point-cloud described previously becomes useless if the observation cannot be linked with its position in the environment. While we previously explained that certain sensors provide estimations of position and/or orientation, solving localization is a standalone research subject. In the upcoming content, the term “pose" simply means position and orientation.

Commonly, the localization is the problem of estimating the robot’s pose in an external reference frame, from sensor data, using a map of the environment [29]. Alternatively, the problem can be reformulated to estimate at the same time the robot’s pose and the map,

making it a SLAM problem (Simultaneous Localization and Mapping). In this section, we briefly introduce the challenges linked with localization, and its problem formulation.

It is important to note that this research does not contribute to the field of localization. However, understanding the associated challenges is essential for comprehending the content of this manuscript. For readers interested in an in-depth study of localization, [29] would be the authoritative reference book.

Frames

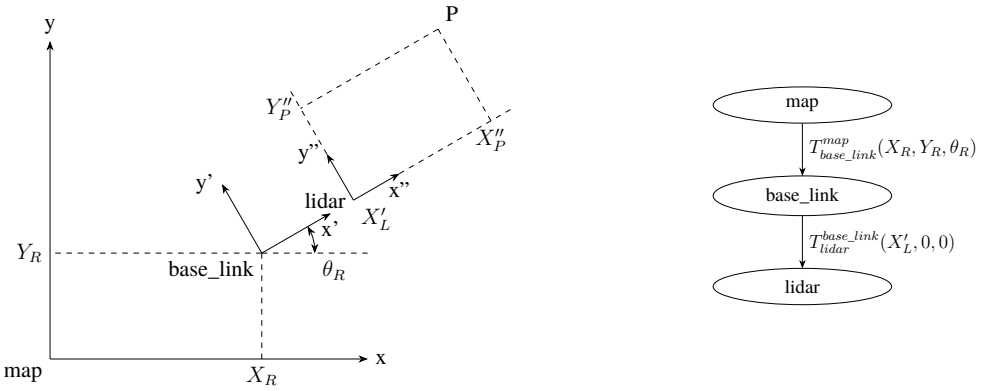

Figure 2.4: Illustration of the concept of frames in a 2D-example. On the left: different coordinate systems, or frames. The external reference frame is called *map*, the robot frame is called *base_link*, and the Lidar frame is called *lidar*. On the right, the tree structure storing the transformations between frames.

When considering localization, the primary concept to grasp is linked to frames. In a robotic system, several frames exist, each defining a different coordinate system. Solving the localization consists in estimating the robot pose, in a reference frame, from observations that may be in different frames. When the problem is formulated in 3D, the pose corresponds to the six degrees of freedom of the system: its 3D Cartesian coordinates, and its three elemental rotations, in the reference frame.

We illustrate this concept in Figure 2.4, with a 2D example. This figure shows different coordinate systems. First, on the bottom left, the *map* frame represents the global coordinate system. Second, the *base_link* frame corresponds to the current robot pose. In the *map* frame, this pose is the vector (X_R, Y_R, θ_R). To this vector corresponds the transformation between *base_link* and *map*. Next, the *lidar* frame corresponds to the coordinate system in which the point-cloud from the Lidar is expressed. In the *base_link* frame, the lidar pose is the vector (X'_L, Y'_L, θ'_L). In this particular case, $Y'_L = 0$ and $\theta'_L = 0$. To this vector corresponds the transformation between *lidar* and *base_link*. Finally, the coordinates of the point P1 in the

point-cloud are (X''_P, Y''_P) in the *lidar* frame.

In robotics applications, it is convenient to keep track of the relative position of a frame with respect to another one in a tree structure. Indeed, it is likely that *lidar* remains fixed with respect to *base_link*, and that *base_link* will move with respect to *map*. By chaining the transformations between frames, we can, for instance, easily recover the position of the point P in the *map* frame.

Problem formulation

If a sensor could offer a highly accurate estimation of the robot pose, localization would not be such a challenging task. However, even if such a perfect sensor were to exist, a new challenge would emerge: how could we then be certain of the robot's position with respect to its environment with absolute certainty ? Moreover, what if we introduced another complication, like the presence of humans moving in the robot's surroundings ?

In mobile robotics applications, the fundamental reality is that nothing is known with absolute precision. Every measurement, from any sensor, carries some noise. Additionally, every action the robot takes introduces its own noise. When a command is transmitted to the robot's actuators, the actual movement of the robot always slightly deviates from the anticipated trajectory. This can be attributed to sensor noise, such as in the wheel encoders, but more broadly, it comes from the challenge of precisely modeling interactions with the environment. This, once again, contributes to the noise inherent in the robotics system.

For these reasons, probabilistic algorithms are the fundamental algorithms in mobile robotics, as expansively detailed in [29]. The key idea of such algorithms is to represent information by probability distributions over a whole space of possible hypothesis. In the case of localization, the initial belief (the initial robot's pose estimate) is represented by a probability density function over the space of all locations. Every new sensor measurement (observation) and every new movement of the robot (action) update the previous belief. Generally, with enough actions and observations, the probability mass is moved to a single location, and the robot's localization is known with a good confidence.

This problem formulation is a state estimation, and this description depicts a Bayes Filter. The two more widely used localization algorithms in mobile robotics derive from this algorithm: the Kalman filter and the particle filter, which we simply mention here. Once more, our focus here is on introducing the localization challenges, and we encourage readers to consult [29] for comprehensive explanations.

2.2 Mapping

2.2.1 Fundamentals of mapping

A map is a representation of the environment. The selection of a specific map representation depends on various factors. First and foremost, if the map is intended for robot navigation, it must be computed online on the robot, which introduces a significant computational complexity limitation. Conversely, if the map is computed offline, this limitation is no longer a concern. Secondly, the map's intended purpose and the type of information it needs to store are essential considerations. Lastly, does the map exclusively stores static objects? Different map representations are available based on these considerations. We will now describe a few examples, focusing solely on static maps.

Firstly, a continuous map is an exact decomposition of the environment. It compiles a list of all objects along with their location. Typically, to manage computational costs, the objects are selected and abstracted. For instance, line segments are extracted from sensor data, and their coordinates are stored, as presented by [31]. These types of maps are commonly employed in indoors applications. Secondly, a discrete map is a discretization of the environment. These maps typically assume that geometry is the most salient feature of the environment. Such maps are prevalent in various mobile robotics applications. We employ them in this research, and we will delve into their details in the following section. Thirdly, topological maps are graphs that define nodes and the connectivity between nodes. GPS navigation relies on this map type, where a node signifies an intersection, and the connections between nodes represent feasible trajectories. These maps also find widespread application in robotics, due to their capacity to enable fast search algorithms within them. More recently, research explored semantic information incorporation into the map representations. For instance, in a topological map, each node is assigned a semantic label like "grass" or "asphalt", as shown in [42].

In the upcoming paragraphs, we will elaborate on a specific type of discrete map known as the occupancy grid. We will then showcase different challenges related to mapping, which are essential for a thorough understanding of this manuscript.

Occupancy grids

As previously mentioned, achieving optimal navigation planning stands as one of the most prevalent objectives for building a map. In other words, the intention behind the map is to plan the robot trajectory, from its current position, to a goal. In such a case, the map is directly linked to how the trajectory will be planned. The most prevalent map in this case is the

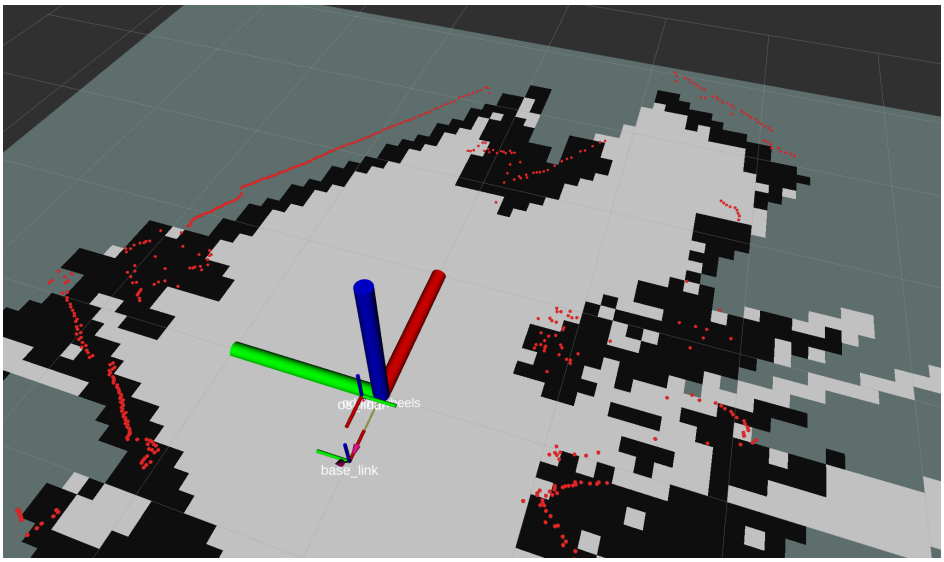

Figure 2.5: Example of a 2D occupancy grid map, mapping the same place depicted in Figure 2.2. Black: pixel corresponding to occupied space, light-grey: free space, dark-grey: unknown space.

occupancy grid, a discrete map. The occupancy grid is a 2D-map, an image, representing the plane the robot will evolve in. Every cell in this map, each pixel in the image, encodes the necessary information to compute a safe navigation plan. The simplest form of occupancy grid contains ternary information: each pixel is either free, occupied or unknown. A slightly more advanced form of occupancy grid encodes the occupancy likelihood of each pixel. When the robot is equipped with a Lidar, every point-cloud provides information. Basically, we know that every return point corresponds to occupied space, and the ray between the robot and the return point corresponds to free space. Behind the return point, we do not have any information. Building the occupancy grid consists in accumulating the robot's observations in time.

Figure 2.5 provides an example of an occupancy grid map. This figure showcases some mapping features described previously. Firstly, this map is an image, projected on a plane. The dark-grey areas on the top corners are outside the map. Secondly, the figure shows the current projection of the 3D-Lidar on a horizontal plane (red dots). Thirdly, on the map, we can see the occupied space where those points are projected on the ground (black), the unknown space behind (middle-grey), and the free space between the robot and the points (light-grey). Secondly, this figure shows the results of the ray casting operation. Some previous returned Lidar points have marked the space empty only between the return point and the robot, leaving unknown space around the rays (middle-right of the figure). Finally, although this is not obvious in the figure, it also displays the different frames involved in the mapping.

The map is built in the frame represented by the thick axes. The small axes in the figure are the lidar frame and the base_link frame (the current robot pose in the map).

Challenges with mapping

The core of the mapping challenge comes from the fact that the robot is moving. The robot's observations are always in the sensor frame. The very frame attached to the sensor. As shown in Section 2.1.2, that frame is easily transformed into the robot frame, because the pose of the sensor with respect to the robot is known, even though some uncertainty remains. The challenge comes from transforming the robot's frame into the map frame, the only fixed frame in time. This challenge is the localization introduced previously. Every robot's observation, in the robot's frame, updates the map. Every error in the estimate of the robot's pose in the map leads to errors in the map's updates. The uncertainty in the robot pose produces uncertainty in the map, as shown in [20].

An additional issue is linked to the sensor itself. A Lidar is not exempt from measurement noise. Furthermore, the noise level tends to increase in natural environments, due to several factors. For instance, from a laser's perspective, trees can behave as semi-transparent structures, thereby inducing errors in the measurements. The reason is that when laser-rays, light cones in practice, reach a small object, like a branch, often only a portion of the energy is reflected. This causes multiple echoes, which are a common source of inaccurate distance readings. For instance, when measuring distance to a small branch, depending on the amount of vegetation, the intensity of each echo may vary. The distance reading, associated to the highest intensity echo, may be the small branch, another branch behind, or some larger structure behind the tree, for example. A comprehensive analysis of this phenomenon is provided by [76]. Two other causes of errors in laser measurements, that are particularly abundant in natural environments, are linked to the distance to the objects and the incidence angle: the error increases with each of them [46].

Figure 2.6 illustrates the effect of noise on the map. Rows A and B show the effect of noise only in the localization. Row C depicts the effect of noise on the Lidar only on the map. Finally, row D exemplifies the effect of a combination of noise of the lidar and noise on the localization. Looking at column (0), where the localization is perfect, we can see that the addition of noise in the sensor blurs the cross extruded shape, but much less than the addition of noise in the localization (column (2)). The examples in Figure 2.6 are not maps, but simply accumulations of point-clouds in time. Regarding maps, especially occupancy grids, it is worth noting that every point-cloud contributes to the map. Indeed, every return point in the point-cloud, with its noise, is transformed into the map frame, with the noise in the transformation, and used to update the map. In that case, the occupancy likelihood of

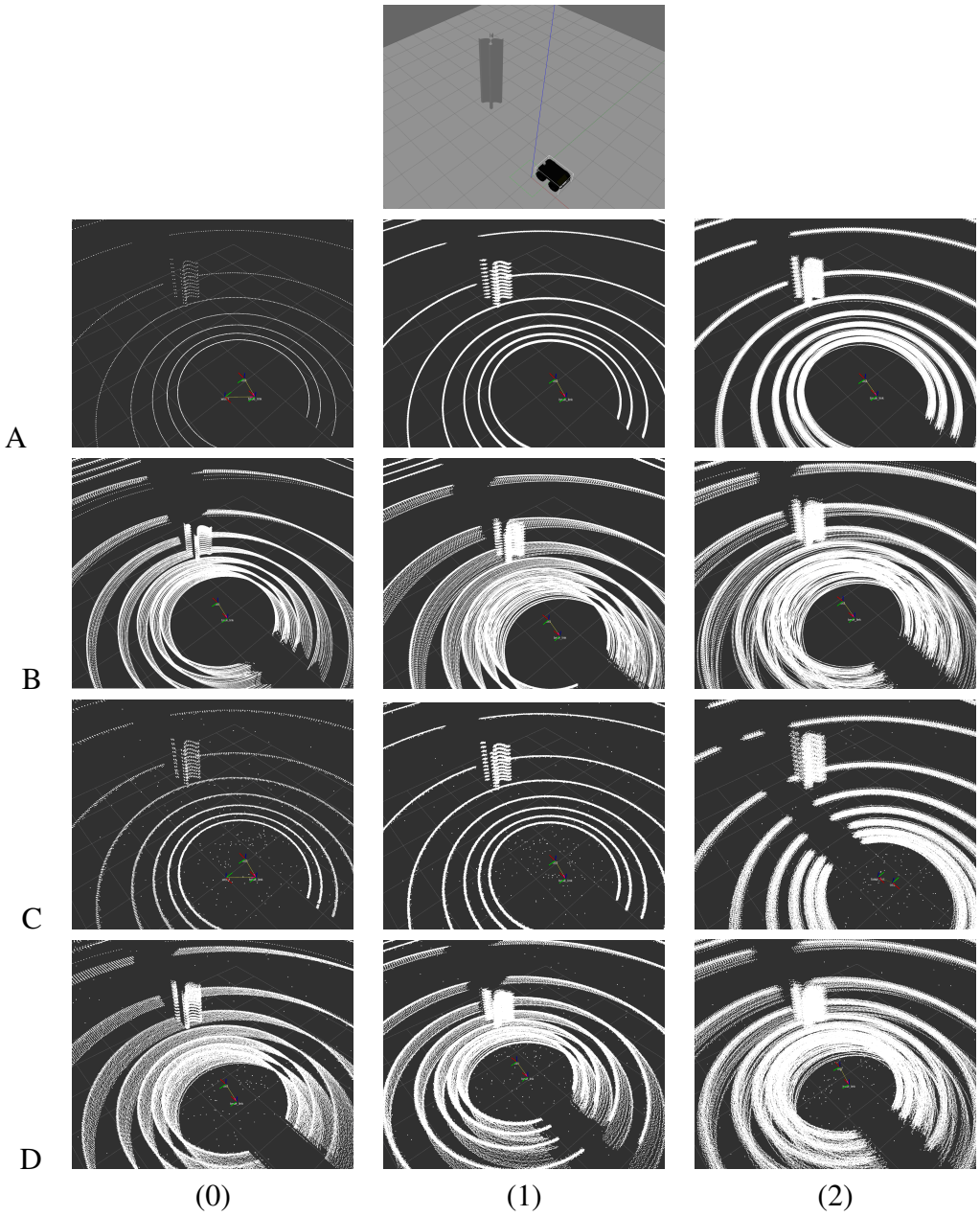

Figure 2.6: Illustration of the effect of noise on the map, on a toy-case. The noise is applied either on the localization, on the sensor, or on both. Top: the simulation: a husky-robot facing a cross-extruded shape. All the other figures display the point-clouds accumulated for five seconds in a given map frame. The frame is either the perfect localization from the simulation (0), or a noisy localization, where we apply a Gaussian noise to (0). We display two level of noise: (1) and (2). Rows A, B: the Lidar sensor is perfect. Rows C, D: the Lidar sensor is noisy. Rows A, C: the robot is not moving. Rows B, D: the robot is moving. If we focus on the cross-extruded shape, we can see that the higher the noise in the localization and / or on the sensor, the blurrier the shape.

the pixel containing the returned point is increased, the occupancy likelihood of the pixels traversed by the laser ray are decreased.

Purpose of mapping

Often, robot mapping is intrinsically linked to the autonomous navigation of the robot. Planning a trajectory within an occupancy grid consists in computing a trajectory from the robot's current pose to the goal, while ensuring it stays in free space and maintains a safe distance from obstacles. This computation relies on the values associated with the pixels of the occupancy grid, that is, the occupancy likelihoods. Optimal planning in a 2D-plane may be considered a solved research problem, and several algorithms exist to compute optimal plans. It is worth noting that we are specifically discussing here global planning, which involves going from point A to point B. The distance from A to B can be large, and in addition to the global plan, the robot actually needs to follow what is called a local plan, reactive for example to moving obstacles, and to be controlled on this local plan. Planning will be briefly presented later in this manuscript.

However, there are cases where the purpose of the map is not the planning. For instance, it may be the need to build a representation of a scene enabling its monitoring, such as in [19]. This is the scenario we consider in this research, as we will see in Section 2.2.3.

2.2.2 Sparse and Unstructured Environments

In this research, we consider a particular type of environment: natural environments. In addition to be potentially large, they combine two challenges: they are both unstructured and sparse.

Unstructured Environments

Let us begin with the lack of structure. In the literature ([57], [1], [20] or [7]) it is customary to distinguish between structured and unstructured environments. Generally, we consider the environment structured if its underlying structure can be approximated by basic geometrical shapes such as rectangular cuboids, cylinders, etc..., unstructured otherwise. Following this, natural environments are considered unstructured, whereas urban areas are more structured. Research often focus on “structured" environments like urban areas, as demonstrated by extensive research conducted on the Kitti Dataset [30].

In unstructured environments, the difficulty comes from the sampling of the surface. The sampling of the surface is the points in the point-cloud that correspond to the object. In a structured environment, the sampling of a surface does not need a high density to recover the

underlying structure. On the contrary, this is not true in an unstructured environment. We illustrate this statement with Figure 2.7. It displays some structured elements: on the left side, a building and in the foreground, on the right side, a lamp and a bench. It also displays unstructured elements: two trees on the right in the background. When looking at the point-cloud of the same scene in the bottom, it seems easy to recover the underlying structure of the building, the bench, or the lamp. We refer here to the ability to perform the task for a human observer, regardless of the feasibility of the task by any automatic method. On the contrary, when looking at the point-cloud of the trees, this is quite different. We can recover the underlying structure of the trunk and the large branches, but nothing in the canopy. The two pictures in the bottom zoom into the two red rectangles: on the trunk, and on the canopy. The zoom level is the same on both pictures. Looking closely at these pictures does not help in the canopy. Some points may correspond to small branches, some other to leaves, we can only guess. The reason lies in the sampling of the surface. If the sampling of the canopy had been higher, we could have been able to recover the structure, but that seems not achievable in a real world setup.

It should be worth noting that the sampling is also directly linked to the density in the output of the 3D-Lidar sensor mentioned earlier. In this example, the point cloud comes from the scanning of the area with a Leica Total Station. A robotic Total Station is a device composed of a highly accurate laser, controlled in angular position with a high accuracy. It allows performing 3D-scans of areas, with a precision and a density much higher than achievable with any embedded sensor.

Sparse Environments

Then, let us continue with the sparsity. The sparsity comes from two sources.

Firstly, it comes from the environment itself. Natural environments tend to be sparse. They contain mostly empty space, where no Lidar returns, ground, and some objects, such as trees, bushes, rocks etc... The objects themselves are also sparse. Except for tree trunks, vegetation is generally sparse. It contains mostly empty space, with small branches, leaves, etc...

Secondly, the sparsity comes from the sensing modality. A 3D-Lidar outputs a sparse point-cloud. For example, the Ouster OS1-16, used in some experiments in this work, outputs 82×10^3 points per seconds in its cylindrical field of view. Another sensor used in this work, the Robosense BPearl outputs 576×10^3 points per seconds in its hemispherical field of view. As a comparison, the depth output of an Intel RealSense D415 can reach 83×10^6 points per seconds, in its rectangular field of view. An Intel RealSense is an RGB-D camera, a camera providing at the same time the RGB image, and the depth of each pixel (the distance to the

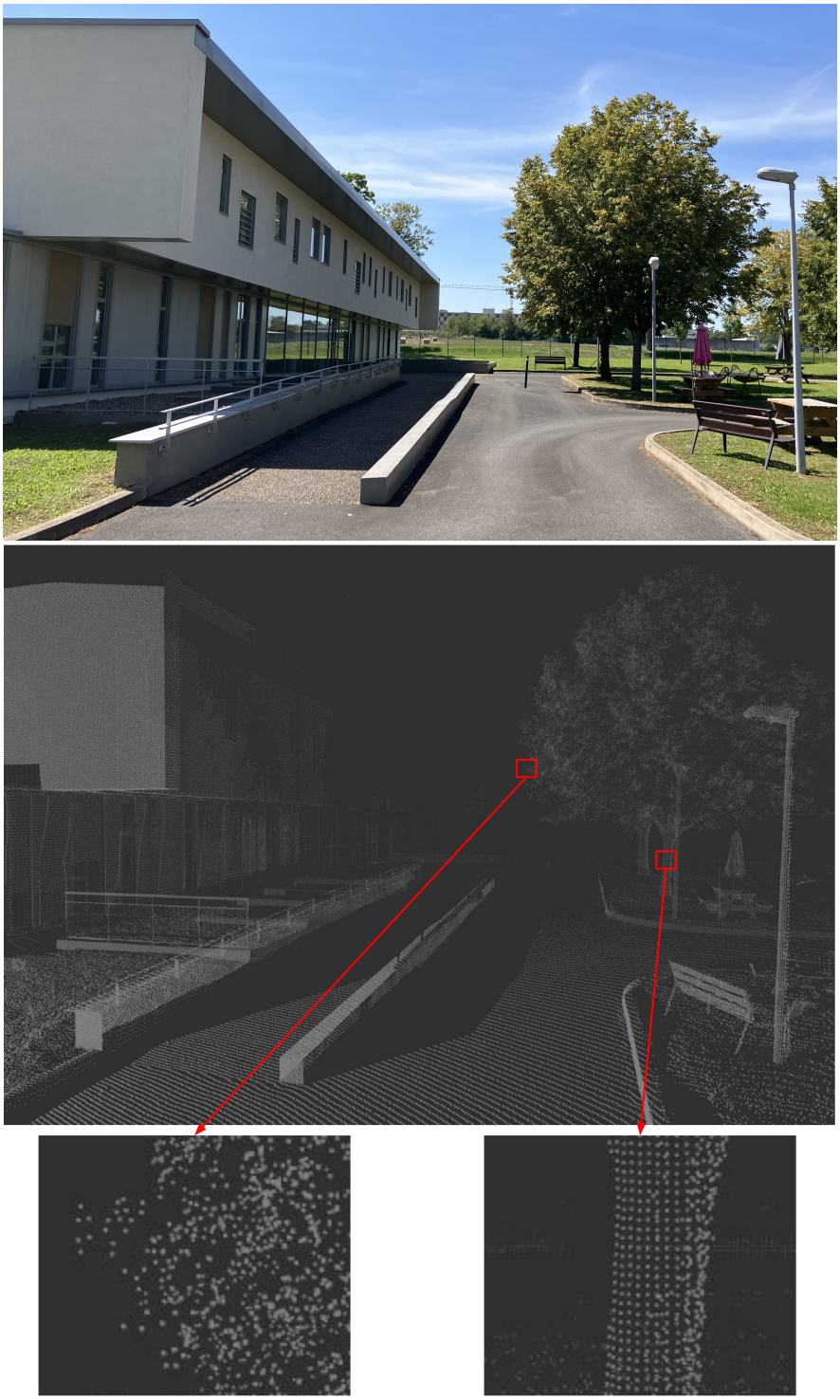

Figure 2.7: Illustration of the structured or unstructured nature of the environment on the sampling of the surface. The middle picture is the point-cloud of the place shown in the top picture, acquired with the Leica Total Station. The bottom pictures highlight the rectangle areas above by zooming in.

sensor). Such sensors are not suited for outdoor applications, mostly because the accuracy of the depth output decreases rapidly with the range, and because they are not robust to other typical outdoor problems, such as sunglare.

As a consequence, the point-cloud representing a natural environment is sparse. That is true even if we do not yet consider the map, but only the accumulation of the robot's point-clouds in the map frame. This resulting point-cloud consists predominantly of empty space, with only points where the 3D-lidar actually hits an object, further complicated by the noise level, in the point-cloud, and in the localization. Figure 2.8 illustrates this difficulty. This figure shows that the point-cloud from the 3D-Lidar is sparse, and it also highlights the difficulty introduced by an error in the localization. It is particularly visible on salient objects, such as the pole in the left. The points from the Lidar does not appear superposed on the point-cloud from the Total Station. This error comes from the localization.

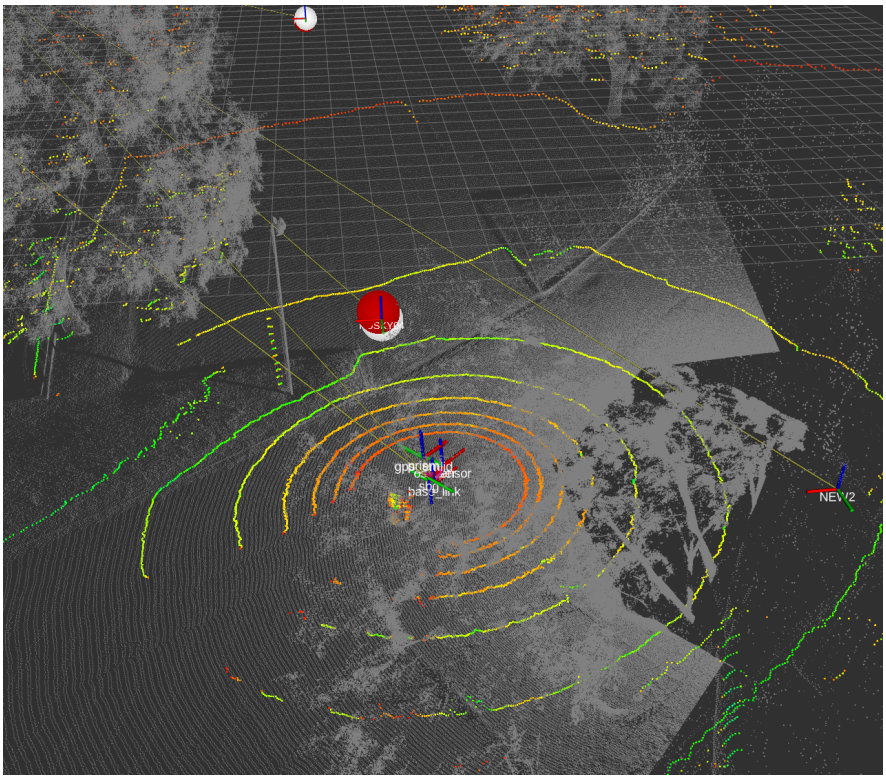

Figure 2.8: Illustration of the sparsity of the data output of the 3D-Lidar. The white dots corresponds to the point-cloud acquired by the Total Station. The colored points corresponds to a single point-cloud from the robot's Ouster-16. The spheres are debugging tools, used to display residual errors in the localization.

2.2.3 Map Representations

In the scenario we consider in this thesis, the purpose of mapping is to build a geometric representation of a scene. As such, the most common maps are volumetric maps, or 3D-grids, and meshes, or 3D-surface maps. We explain both here.

Meshes

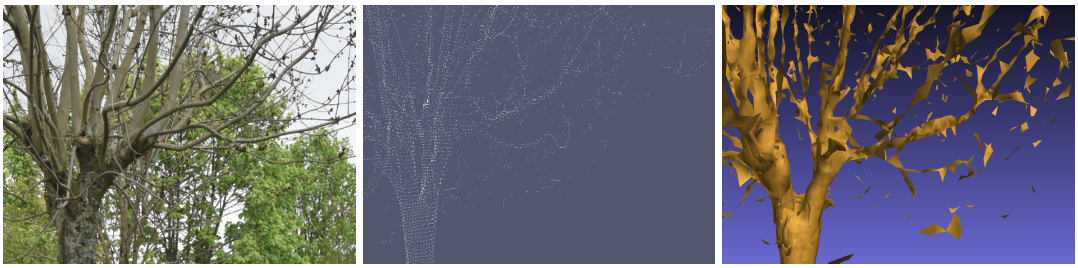

Figure 2.9: Challenges with meshing: (a) the real tree, (b) the point-cloud from the Total Station (c) an example of mesh failing to describe the actual surface of the fine elements of the tree.

A mesh is the description of a surface by connected triangles. Research is prolific on meshing single objects, but much less on meshing large scale environments. Generally, the mesh is build from a point-cloud. We apply an optimization algorithm, and find the connected triangles that best describe the surface. Among the most widely used techniques, we can cite Ball-Pivoting Algorithm [12], Poisson Surface Reconstruction [40] or Delaunay triangulations [21], mainly applied to mesh a single object. Other methods tackle large-scale meshing, such as [10] who includes texture in the mesh to improve its visual aspect, or [19] who build a semantic mesh representation of a large-scale environment with the Las Vegas Reconstruction Toolkit [79]. Recently, other methods have been presented to build meshes with deep learning techniques, such as Voxel2Mesh [78]. In the large scale, sparse, unstructured environments we consider in this thesis, the mesh is unlikely to represent the surface. The reason lies in the sampling problem we highlighted before. In some applications, whereas the mesh may appear visually nice due to its texture, in reality, the geometry does not describe accurately the surface. We illustrate that with an example Fig. 2.9, where the mesh is derived from what could be considered a dense point-cloud, the point-cloud obtained from a Leica Total Station. We see in the middle figure the sampling problem we mention. The trunk is quite clear, but the branches much less. Knowing it is a tree, our mind is capable of interpreting that the points spread on the right are likely small branches, but from a computer perspective, there is little chance that any optimization converges on a mesh that actually describes the surface.

3D-grids

A 3D-grid is a generalization in 3D of the 2D-grid we introduced earlier. Such a map is the discretization of the space into voxels, each voxel containing some information. That information can be:

1. an occupancy map, where the voxels have 3 states: free, occupied or unknown;
2. a probability map where each cell contains its probability of occupancy;
3. or something different, where the voxels contain data such as semantic information.

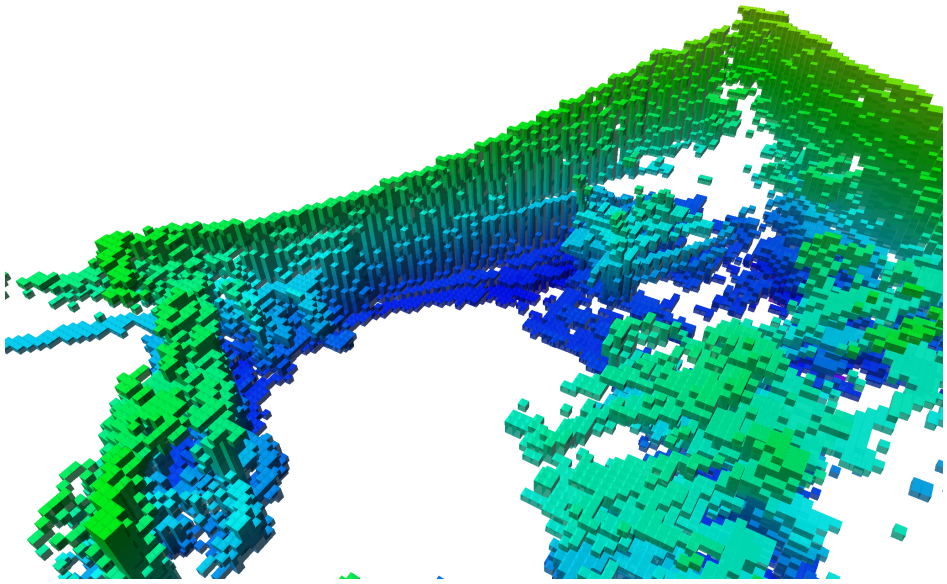

Figure 2.10: Example of a 3D-grid map, mapping the same place depicted in 2.2. Each cube is a 5cm side voxel, representing only the occupied space. The color encodes only the height, to help the visualization.

In this work, we focus on the prevailing map representation for outdoor robotics: probabilistic 3D-grids. The appearance of this map is directly derived from the resolution, that is, the size of an edge of a voxel. Ideally, the lower the resolution, the more detailed the map. Figure 2.10 provides an example of a 3d-grid map. This map is a representation of the lab shown in Figure 2.2. Due to the small vertical field of view of the Ouster-16 3D-Lidar, only a slice of the lab is actually mapped. Also, because of occlusions, the back side of the other robot (top right corner) is not observed, and therefore, not mapped. In this example, the resolution is set to 5cm-side voxels. The resolution should be chosen carefully to be consistent at the same time with the type of objects we want to map in, and the level of noise when building the map, particularly in the robot localization. As an example, if the position of the

robot is known with an $0.2m^2$ uncertainty, building a 3D-grid with a $0.05m$ resolution is not suited and will lead to a highly noisy map.

2.3 Autonomous Exploration

2.3.1 Autonomous Navigation

Before looking into autonomous exploration, a prerequisite is to understand what is autonomous navigation. Autonomous navigation is a high-level task in mobile robotics. It consists of enabling for the robot to move autonomously and robustly in its environment. Autonomous navigation is divided in three components: global planning, local planning and control. Exploration is a higher level task that consists in choosing the next goal. Often, global planning is part of the exploration task, as we will see right after.

First, we focus on the three navigation tasks.

Global Planning

Global planning consists in planning a safe trajectory from the current position of the robot, A, to its target position, B. Planning such a trajectory requires several components. First, we need to define where the robot can or cannot go in the map. If we take the example of a 2D-occupancy grid map, we can assume the robot can navigate safely in free space. Two questions would be: can it navigate safely close to obstacles ? Can it navigate safely in unknown space ? Answering those questions require the computation of an intermediary component called the cost-map. This cost-map is a 2D-grid where each cell is associated to a cost. Planning from A to B would then consist in finding the lower cost trajectory in the set of feasible ones from A to B. A typical cost-map consists in simply expending the obstacles in the occupancy grid with a Gaussian Kernel. Cells close to obstacles would then have a higher cost than those far from obstacles.

More advanced cost-maps would take other factors than simply the distance to obstacles. For example, the cost of a cell may be linked to the slope of the terrain it is associated to. Similarly, each cell could be assigned a traversability score depending on the very nature of the terrain. For instance, asphalt is easily traversable, whereas rocks are not traversable. In the middle, short grass could be traversable, although less easily than asphalt, and tall grass may be traversable, but with some risk, and the cost should reflect the level of risk of planning through each cell. Global planning is finally an optimization to seek for the cheaper trajectory from A to B. Several algorithms allow to compute this best trajectory, among which the more

broadly used are Djikstra, A*, or RRT*. We recommend the interested reader to look for [67] that provide a comprehensive description of the main planning algorithms.

Local planning and control

Although local plan and control are out of the scope of this work, we provide to the reader high-level information about what they are in the context of mobile robots, and again invite the reader to look for [67], a reference book in mobile robots.

Local planning enables the local adaptation of the robot to the global plan. The more obvious reason is moving obstacle avoidance. By definition, it is not possible to plan a path from A to B avoiding moving obstacles. This reactive behavior is what local planning deals with. Local planning is reactive planning to keep executing the general trajectory from A to B, while taking into account local changes.

Control is the final step in autonomous navigation. It consists in computing and applying the commands that will drive the robot to follow the local plan.

2.3.2 Exploration

As introduced before, in an exploration problem, we consider a volume unknown at the beginning. The robot moves into this volume, and at the same time gathers information and reduces the unknown. Crafting an exploration policy consists in defining the rule to choose where the robot should go next. That policy is directly linked to the reason why the robot has to explore the environment. Generally, whether it be only for its navigation, or for other purposes, the map is directly linked to the exploration. Let us assume the map is a 2d-occupancy-grid map introduced before. This map encodes the occupancy likelihood of each pixel, and therefore contains information about occupied space, empty space, and unknown space. Occupied and empty space together are the known volume, the remaining is the unknown volume. From this frontier between the known and unknown volume [81] introduced the concept of frontier points. A frontier point is a point, within the known volume, with at least an unknown neighbor. From this concept derives the first autonomous exploration policy, also called closest frontier exploration. In closest frontier exploration, the next goal is always the closest from the current robot position in the set of reachable frontier points. This decision-making process, that drives the choice of the next goal to visit, is called the exploration policy. The primary objective of a closest-frontier exploration policy is to rapidly reduce the unknown in the map. Another traditional exploration policy, derived from frontier points, consists in randomly sampling the goal among the frontier points rather than choosing the closest. This policy is commonly referred to as random frontier exploration.

In a typical exploration mission, the sequence of actions involves: setting the goal, moving to the goal, updating the map, setting a new goal, and repeating this cycle. However, this behavior is not necessarily linear. For instance, the map can be updated while the robot moves. Similarly, the next goal is often regularly sampled, without waiting for the robot to reach the current goal. This approach is used because, due to map updates while the robot is moving, a goal that is a few seconds old may no longer be the optimal choice. In traditional policies, the cycle is over when the exploration is considered complete. This determination is often based on specific criteria, such as mission time, or achieving a minimal percentage of the volume being known.

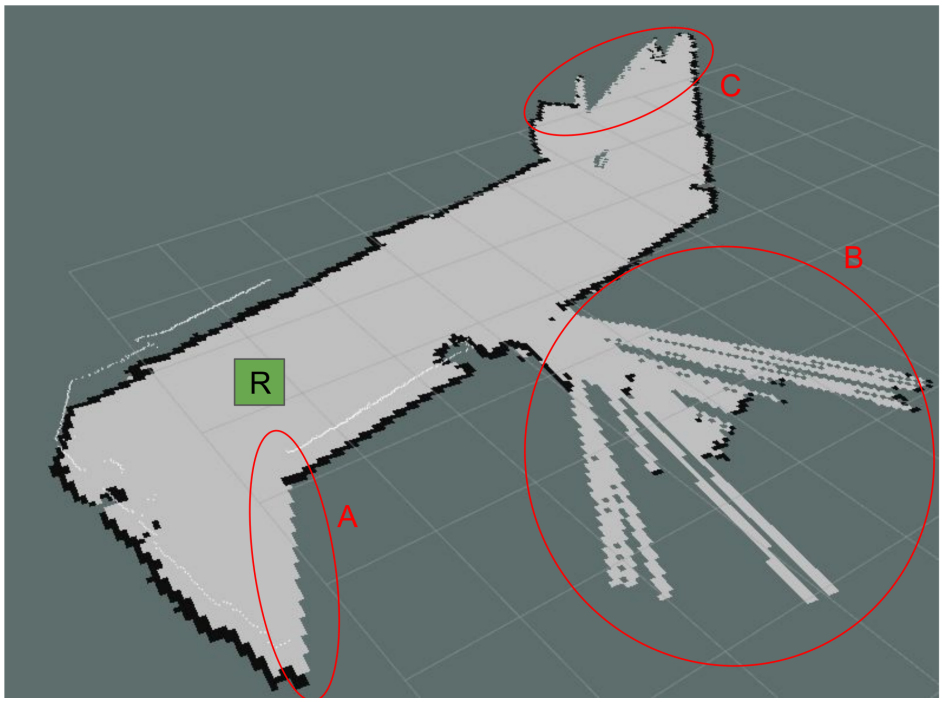

Figure 2.11: Illustration of the concept of frontier points in an occupancy grid. The current position of the robot is the green square. A, B and C are the main clusters of frontier points

Figure 2.11 provides an illustration of the concept of frontier points. The current position of the robot is the green square. A, B and C are the main clusters of frontier points (some frontier points are not in A, B or C). Depending on the exploration policy, the robot may pick a goal in A (closest), in B (highest density of unknown points for instance), or either in C (random for instance).

Whereas out of the scope of this work, it should be mentioned that the concept of frontier point exploration is often extended to multi-agents. Indeed, when the objective is to discover the volume as quickly as possible, having robots cooperate to build a common map is an appealing solution, as proposed initially by [17].

2.3.3 Next Best View

As mentioned earlier, the choice of the next goal depends not only on the target destination, but also on the path the robot will take. During the execution of the path, observations continuously update the map. Because of this continuous update, a new goal is typically chosen during the execution of the path. In this context, the robot's actuation is driven by the objective of maximizing perception efficiency, a strategy commonly referred to as *Next-Best-View* (NBV) or *active sensing*

NBV is an active research topic in various contexts. For instance, [8] selects NBVs based on their "perceptual informativeness" to minimize localization error. The underlying idea is that when the primary sensor is a camera, avoiding featureless areas increases the performance of visual inertial odometry algorithms.

When NBV and exploration are linked, the exploration policy is the function defining a balance between the need to maximize the efficiency of the perception (the information gain), and some cost (for example, the distance to the candidate), to select the best goal for the exploration task. Often, exploration policies are linked to simultaneous localization and mapping (SLAM). SLAM, introduced earlier in this chapter (Section 2.1.2), is an optimization that seeks at the same time to find the map, and the poses, or history of poses, of the robot. When exploration and SLAM are linked, the gain is often formulated to reduce both map and localization uncertainty, as in [70]. Beyond SLAM, the objective of the NBV selection, within the context of exploration, often revolves around finding the path that maximizes space discovery, as shown in [13] for instance.

In contrast, in this work, we not only seek to reduce the unknown in the map, but we also consider localization as an input to the mapping problem. In our research, map quality is a key feature of the overall objective. The central question we address in this thesis is: "how can we select the Next-Best-View to enhance map quality in a natural environment ?" While some methods focus on structured environments (discussed in Chapter 6), our novelty lies in developing a method effective both in structured and unstructured environments.

3

Framework: local maps and experiments

The initial phase of this work involves setting up a framework that lays the foundation for our primary goal: creating an exploration strategy for possibly large-scale natural environments to enhance map quality. As we delved into the problem, we quickly realized that a localized approach was essential. This applies not only to the exploration strategy, but also to how we assess quality. To address this, we decided to divide the environment into smaller cuboid regions, each containing key local information like map quality and viewpoint statistics. This localized perspective makes the data we consider more relevant and meaningful.

The main contribution presented in this chapter is our methodology for this localized approach. We explain how we extract intersecting cuboid regions from both the real (or simulated) environment and the robot's interactions with it. These cuboids form the basis for the work discussed in the upcoming chapters.

Additionally, we introduce our experimental framework, which includes simulations and real-world experiments. Following the conventions in robotics, we use ROS [1] (Robot Operating System) as our middleware. We prioritize the use of standard ROS packages in components we do not developed ourselves. Furthermore, we open-source all the ROS packages developed as part of this work.

3.1 Mapping

3.1.1 Map building

As introduced in the previous chapter, the map representation used throughout this work is the probabilistic 3D-grid. A 3D-grid is a discretization of the volume into voxels. In probabilistic 3D-grids, each voxel encodes the occupancy likelihood of the volume it corresponds to. In this work, we do not develop any new map building framework. Instead, we use a common 3D-mapping framework in robotics: Octomap.

Octomap [35] is a method to build and store an octree instead of a 3D-grid, saving memory and computation. An octree is a tree data-structure, where each element, a node or a leaf, has at most eight children. That data-structure is highly efficient to partition 3D-space by recursively subdividing it into octants, as illustrated in Figure 3.1. The higher the depth in the octree, the smaller the resolution of the leaf. An interesting feature in that data-structure is that it allows "pruning", that means removing all the children of a leaf to reduce the resolution when high resolution is not necessary. The open-source ROS Octomap library [2] implements the complete probabilistic map building process. Except for the fact that the map is an octree,

[1] `https://www.ros.org/`
[2] `http://wiki.ros.org/Octomap`

the map building is similar to what is presented in Chapter 2. The map is constructed in a given map frame, and every point-cloud in the Lidar frame updates the map. For each point in the point-cloud, a ray-casting operation is performed. Between the robot and the returned point, the probability of occupancy of the leaves in the octree is decreased. The returned points are updated as occupied, their probability of occupancy is increased. A thorough update process has been implemented to be robust to noise in the Lidar's point-cloud. Moreover, the leaves in the octree encode the likelihood of occupancy, or emptiness, only if a point has been observed. Therefore, the octree encodes the unknown volume, i.e. the volume represented by absent leaves, a feature that can be useful in different robotics applications, such as exploration.

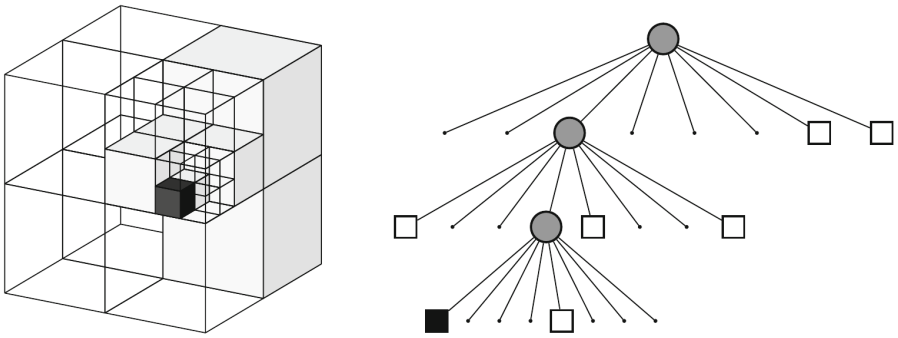

Figure 3.1: From [35]. Example of an octree storing free (shaded white) and occupied (black) cells. The volumetric model is shown on the left, and the corresponding tree representation on the right.

Several methods provide means to build 3D-grids from 3D-Lidar point-clouds. For instance, Voxblox [54] builds 3D grids where the truncated signed distance field (TSDF) is stored for each voxel. Although that method is developed to produce high quality maps, it tends to struggle when the environment size increases. Other methods have been proven efficient to build large 3D-grids, such as Octomap. Even though Octomap might not be the most efficient, as shown in [41], it is still one of the most widely used methods.

Figure 2.10 in Chapter 2 provides an illustration of a 3D-grid build with Octomap. Since it is not possible to illustrate a 3D-grid containing probabilities, it simply displays a thresholded 3D-grid map. Only the voxels whose occupancy likelihood is above 0.8 are displayed.

In this work, we add two features to the Octomap software. The first one is the ability to save the map directly inside Octomap. The original behavior sends the map through the ROS network to a saver. The maps we build in this work are generally large, and this behavior tends to fail. The second one is the ability to work directly on the recordings (called *bags* in ROS). In the original behavior, every time a new point cloud is received by Octomap, it is added to the octree. This addition may take some time, and the point clouds received during

the operation are lost. In our work, we want a map that reflects the robot's observations, regardless of the computation power of the machine building the map. Our modification ensures that all point-clouds from the bags are processed. This modified version of Octomap is available on github [3].

3.1.2 Local Maps - cuboid regions

In our research, a common approach in different methods we propose consists in considering local regions of the map. Given our 3D-grid representation, we choose to define these areas as cuboid sections of the map. This section explains how we extract these cuboid regions, and this process serves two main purposes. Firstly, it allows us to assess map quality locally, as will be discussed in Chapter 4. Secondly, it helps us calculate statistics from the observation viewpoints for these cuboid regions, as will be detailed in Chapter 5.

To proceed, we define a common framework, consisting of the two fundamental elements:

- the reference frame of the experiment: R_f,
- the size of the cuboid: the resolution *RES*.

With these parameters defined, we discretize any given volume into a 3D-grid, of resolution *RES* and within the reference frame R_f. Within this 3D grid, we refer to a specific local region as a cuboid, designated by the notation C.

Ground-truth and reconstructed 3D-grids

When evaluating the quality of the map, we face two main challenges. The first one lies in the transformation of different map representation into comparable ones. The second lies in the precise alignment of the comparable map representations.

Let us begin with the map representations.

Reconstruction As explained before, the map we build from the robot's observations is in reality stored as an octree. The transformation of this octree into a probabilistic 3D-grid requires some processing. Firstly, we create the octree, a full probability map with Octomap in the reference frame R_f, and with a given resolution *res*. From this octree, and its bounding-box in R_f, we initialize a 3D matrix as unknown space, i.e. values of 0.5, corresponding to the equal probability of the voxel to be occupied or empty. We then iterate on all the leaves in the octree. For each leaf, we set the probability of its associated voxels in the 3D matrix to

[3] `https://github.com/stephanie-aravecchia/octomap_mapping.git`

the probability of the leaf (the unknown space is implicitly described in Octomap with absent leaves). We finally obtain the probabilistic 3D-grid we call “reconstruction" in this work. This reconstruction dataset, $\mathcal{D}_{\text{rec}}$, is a 3D matrix of size (h_2,w_2,n_2), with a voxel resolution *res*. Its associated volume in space is the bounding-box B_{box_2}, in the reference frame R_f. It should be noted that we refer to different resolutions in this section. *res* is the resolution of Octomap and of the reconstruction 3D-grid. *RES* is the resolution of the cuboid. This method assumes $res < RES$.

Ground-Truth To later measure the map quality, we need a reference map. This reference map is called “ground-truth" in this work. Here, we need to distinguish simulation and real world experiments. In simulation, the “ground-truth" is a mesh. Indeed, when running the experiment in simulation, the simulated point-cloud corresponds to the computation of rays between the center of the simulated Lidar, and a collision surface. In our experimental framework, this collision surface is the mesh of the simulated world. The transformation of the mesh into a 3D-grid is done as follows. We first slice the mesh with horizontal planes, with a vertical space of *res*, our 3D-grid spatial resolution. In each plane, we then calculate the intersection of the mesh and the plane, and we store it in a 3D matrix of voxel size *res*. Each voxel on the intersection is occupied and has a value of 1.0, all the remaining voxels are empty with a value of 0.0.

In real world experiments, it is not possible to acquire the ground-truth. Nonetheless, we can approximate it by scanning it with a Leica Total Station, as we will detail later. This scanning provides a high-resolution point-cloud. To transform this point-cloud into a 3D-grid, we first construct a 3D matrix of voxel size *res* containing zeros. We then iterate on the point-cloud, and for each point, we set the value of the corresponding voxel to 1.0.

Whether it be from simulation or real world, we now have the ground-truth 3D-grid. This ground-truth dataset, $\mathcal{D}_{\text{gt}}$, is a 3D matrix of size (h_1,w_1,n_1). Its associated volume in space, in the reference frame R_f, is the bounding-box B_{box_1}.

Intersecting cuboids

To later enable the comparison of the local reconstruction and the ground-truth, we need to extract intersecting cuboids from both datasets. To do so, we load the intersection of both datasets, $B_{\text{box}_1} \cap B_{\text{box}_2}$, in two 3D matrices, M_{gt} and M_{rec}. A voxel v_{gt}^{ijk} from M_{gt} corresponds to the same volume in R_f than v_{rec}^{ijk} from M_{rec}.

A cuboid region is a group of $n \times n \times n$ voxels in each 3D matrix, and is noted C_{gt} or C_{rec}. Those cuboid regions are in fact large voxels of size *RES* $= n \times$ *res*, in the 3D-grid

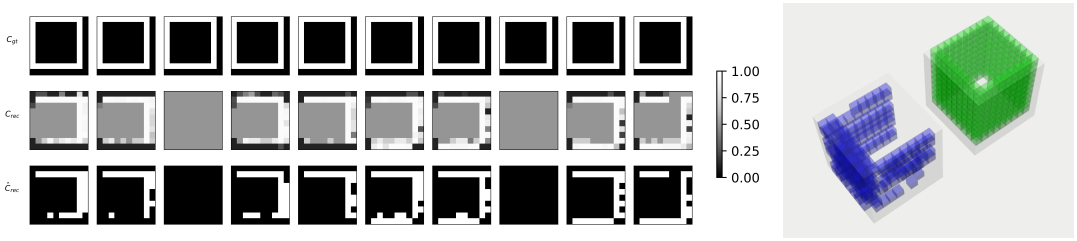

Figure 3.2: Visualization of a cuboid (10x10x10 voxels). On the left part of the figure, each row of images correspond to a single cuboid, each column in the row to a slice of the cuboid. The color encodes the occupancy likelihood, from free space in black, to occupied space in white, as shown in the colorbar. The grey corresponds to the unknown. The first row is the ground-truth cuboid, C_{gt}. The second row is the reconstruction cuboid, C_{rec}, encoding the occupancy likelihood. The last row is the binary version of C_{rec}: $\hat{C}_{rec}$, where the voxels whose occupancy likelihood is above 0.8 are set to occupied, the others to empty. The right part of the figure displays in green C_{gt}, in blue the thresholded $\hat{C}_{rec}$.

$B_{\text{box}_1} \cap B_{\text{box}_2}$, in R_f. Each element in C_{rec} and C_{gt} stores the occupancy likelihood of its corresponding voxel of size res in R_f. Fig. 3.2 shows an example of a cuboid where $n = 10$.

3.2 Experimental Framework

3.2.1 Simulation Framework

This section explains how we generate the simulated worlds we use in simulation. The simulations are run within the ROS framework with Gazebo and a Clearpath Husky robot equipped with a 3D Lidar Ouster OS1-16 (16 planes of 512 points). The simulation framework, including the code to generate our environments, is available on github [4].

World Generation

We generate randomly several environments. Each environment is a plane of dimension $60m \times 60m$ on which we place assets with a Poisson Cluster Point Process, to reproduce the natural spatial distribution of trees.

To populate the simulated worlds, we chose four types of assets, each type of asset presenting a distinct level of complexity in terms of reconstruction. We began with a simple structured shape: a rectangular cuboid. Next, we created a cross-extruded shape, designed to emphasize occlusions in the xy plane. Following that, we introduced a helicoidal cone, capable of producing occlusions in both the xy plane and the z dimension. Lastly, we opted for

[4] `https://github.com/stephanie-aravecchia/unstructured-env-simulator.git`

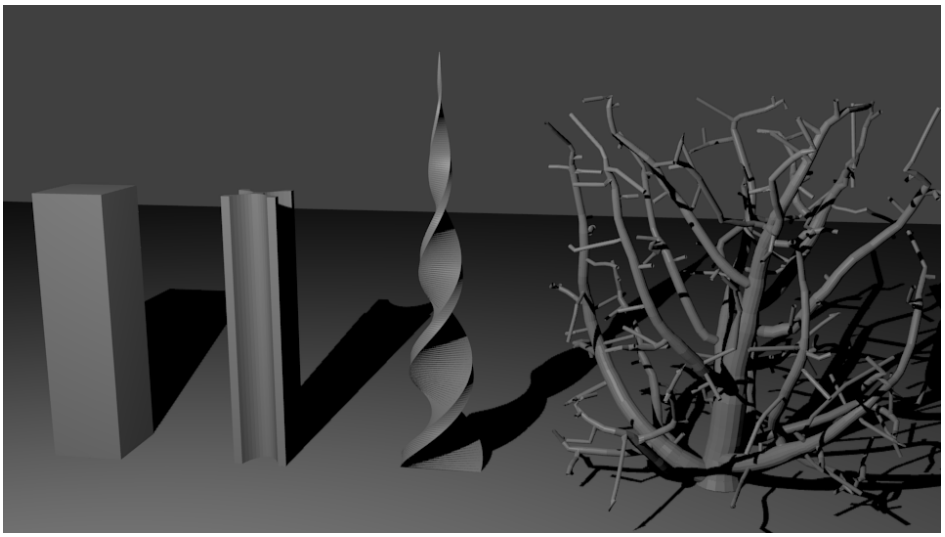

Figure 3.3: Illustration of the different assets used in simulation: rectangular cuboid, cross-extruded shape, helicoidal cone, simulated tree.

simulated trees, representing the most unstructured assets, with occlusions occurring in all directions. Our library includes 15 different tree models, generated using a space colonization algorithm [5], mimicking winter bare trees. Due to computational constraints in Gazebo, these trees are generated without leaves. Finally, when populating the simulated worlds, we apply random factors on the assets' dimensions and orientation. The four types of assets are illustrated in Fig. 3.3.

With the same point process generation, we create four different synthetic environments, one with each type of assets. With the open-source Blender software [6], we create a single shapefile (.stl) for each environment. This ensures that the mesh we slice to obtain the 3D-grid ground-truth is the same used in Gazebo to infer the collisions of the Lidar, hence to construct the map.

We generate 24 environments (3 different spatial distributions with 4 different types of asset, and 2 level of sparsity in the assets). Figure 3.4 shows an example of a same spatial distribution with four different assets, in a sparse environment. As we can see in this figure, the simulated world is enclosed by vertical planes. This arrangement is due to the simulation itself, where there is typically nothing outside the defined world. As a result, many Lidar points do not encounter any obstacles beyond the defined boundaries. Because of the sparsity in the provided example, many Lidar points do not encounter any obstacles inside the boundaries either. A Lidar ray that does not return does not update the map. Consequently, the map remains largely unknown rather than being updated as empty space. In a real-world

[5] `https://github.com/dsforza96/tree-gen`
[6] `https://www.blender.org/`

scenario, this behavior is unlikely. To address this discrepancy, we introduce vertical planes to the simulation environment. This adjustment ensures that when the Lidar observes empty space, it is correctly updated as such, aligning the simulation with real-world behavior.

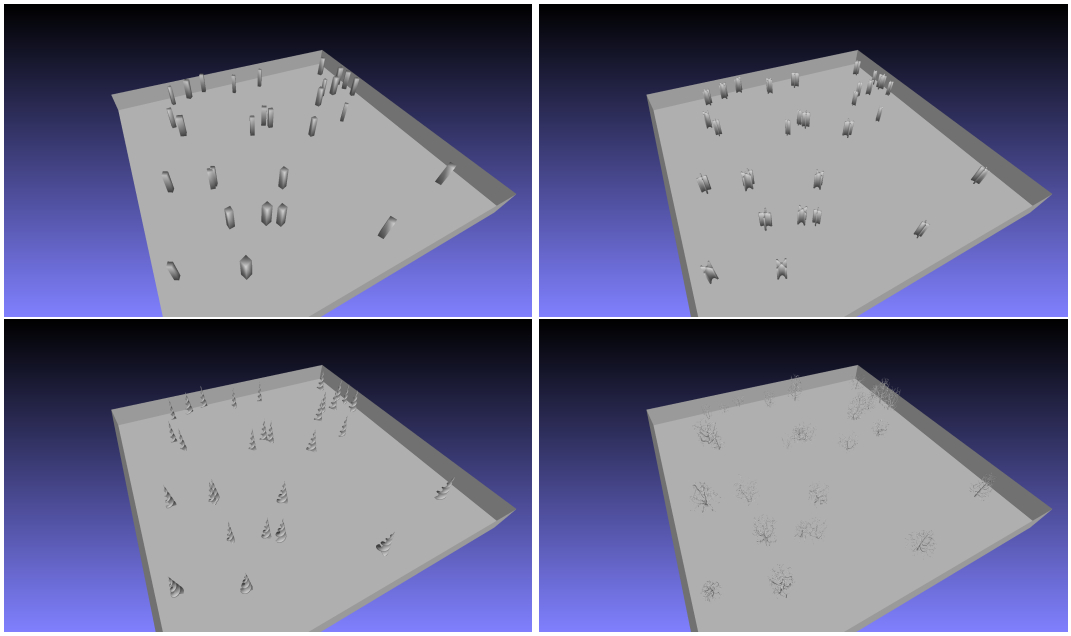

Figure 3.4: Illustration of four simulated words where the four types of assets share the same spatial distribution. In this example, the number of assets is set to create a sparse environment.

Simulated Noise

To make our simulation closer to the real world, we add noise both on the Lidar and on the robot localization.

Lidar Noise

The default noise on the Lidars in Gazebo is a Gaussian noise with a constant standard deviation applied to the range measurements. While this might be adequate for some research, in this research, it is crucial for this work that the simulated Lidar closely mimics the behavior of a real Lidar in outdoor natural environments. To achieve this, we apply a mixture of noise models to the ideal Lidar from Gazebo. Empirically, we found the mixture of noise models shown in Table 3.1, applied on each point-cloud, produced an output visually similar to the real point-clouds, with the same Lidar, in natural environments. For each point in the point-cloud, we compute the noisy range r as indicated in Table 3.1, with r_0 the perfect sensor reading from Gazebo, and r_{min} the minimum range of the sensor.

Ratio of points	Noise formula
96%	$r = r_0 + N$, with $N \sim \mathcal{N}(0,\, 0.008^2)$
3.9%	$r = r_0 - \|\|N\|\|$, with $N \sim \mathcal{N}(0,\, (0.01 \times r_0)^2\|\|$
0.1%	$r \sim U(r_{min}, r_0 + 0.03)$

Table 3.1: Noise mixture applied to the Lidar range

Localization Noise

Since the quality of the reconstruction is directly linked to the localization of the robot, we incrementally add noise in this localization. For each noise level, we build a different probabilistic map with Octomap. To obtain the different noise levels, we use the perfect localization from Gazebo on which we apply a Gaussian noise on the position and on the orientation. The noise levels considered in this study are presented in Table 3.2.

	standard deviation	
Noise level	on position (meters)	on orientation (radians)
0	0.000	0.000
1	0.005	0.005
2	0.050	0.010

Table 3.2: Noise levels in simulation

Errors in the orientation estimation lead to large errors in the position of far away points (i.e a 0.01 radian error in the orientation leads to a 0.20m error in the position of a 20m distant point in the Lidar point-cloud). We consider these three noise levels are representative of various precision levels achievable in real-world applications. It is worth noting that in real-world scenarios, the Gaussian assumption regarding localization may not always hold true. Factors like drift or the non-Gaussian noise often associated with GPS can introduce complexities. However, we state that these considerations do not invalidate the findings of our work, given that we also evaluate it through real-world experiments.

Experimental Simulation

A simulated experiment, illustrated in Fig.3.5, consists in loading one of the environments described previously in Gazebo, and have the robot interact with this environment.

In this thesis, we consider different sets of experiments, as follows. In all the experiments, we generally build the reconstruction with a sensor range is set to 20m, the map resolution to *res* $= 0.1m$, and the cuboid resolution to *RES* $= 1m$. When the robot navigates towards a goal, the planning and control is done with a classic ROS package for navigation: move_base.

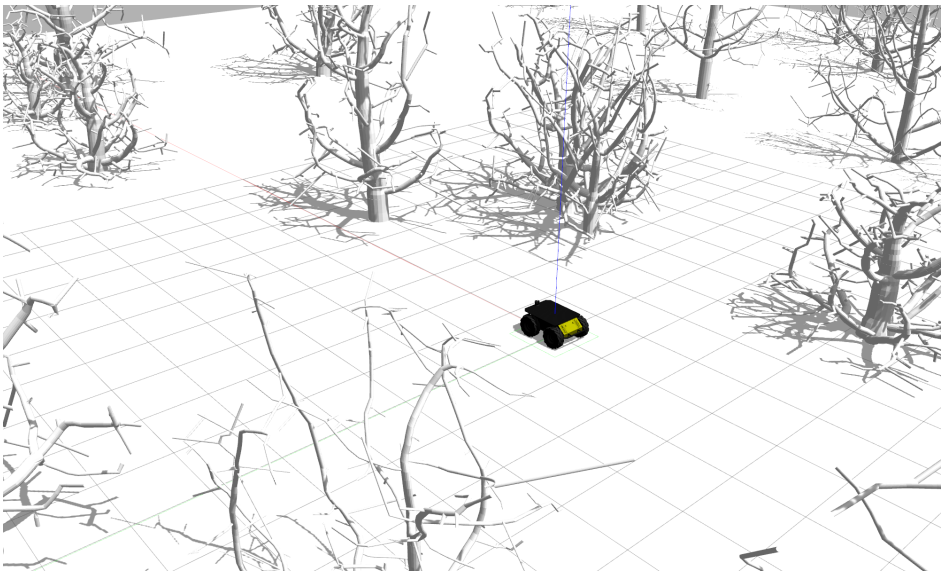

Figure 3.5: Illustration of an experiment, with the Husky robot at the center in an environment containing only trees.

We design various sets of experiments to highlight different challenges related to either mapping or exploration. When focusing solely on mapping challenges, we aim to illustrate how the type of world can make mapping related tasks difficult, as discussed in Chapter 4. Conversely, when examining the relationship between map quality and viewpoint statistics, we want to avoid introducing biases in the viewpoints, as discussed in Chapter 5.

To begin, we categorize our experiments into two main sets:

- **Teleportation Experiments** In these experiments, we teleport the robot into the environment, accumulate point-cloud data in Octomap, and then teleport it again. This process is repeated for 100 iterations.

- **Navigation Experiments** Here, the robot autonomously moves toward a goal, using move_base, and all the observations made during the execution of the path update the map in Octomap.

Next, we design different types of experiments based on what we aim to evaluate:

- **Waypoints Experiments** To eliminate potential bias from different trajectories that could affect map quality, we conduct waypoints experiments. The robot follows a predefined list of waypoints, ensuring a consistent trajectory for all experiments. This approach is a specific navigation experiment, where each waypoint serves as a new navigation goal.

- **Straight Lines Experiments** These experiments aim to highlight the impact of viewpoint diversity on map quality with a visual example, from two different robot behaviors: FRONT and LAT. The robot either drive directly toward an object (FRONT) or alongside the object (LAT). To ensure straight-line trajectories, we controlled the robot directly, bypassing move_base for these specific experiments.

- **Random Experiments** When evaluating how viewpoint statistics influence map quality, we try to avoid trajectory-related biases, where one trajectory might result in more diverse viewpoints than another. For this purpose, we run teleportation experiments, where the goal is randomly chosen.

- **NBV Experiments** Finally, to assess different policies, we conduct teleportation experiments. The goals are selected based on the evaluated policy. Our approach, as explained in Chapter 6, involves evaluating our policies by selecting the Next-Best-View one step ahead, rather than for the entire trajectory to the goal. This aligns better with the teleportation experiments than with the navigation experiments. Additionally, using teleportation experiments speeds up simulations and consequently facilitates data collection, a necessary step in model training.

3.2.2 Real World Experimental Framework

The experimental site is located outside a campus and encompasses a car park area of approximately $5400m^2$, surrounded by a park with trees and bushes, some of which are also situated within the parking lot. Fig. 3.6 illustrates this experiment. Although it is not possible to acquire the ground-truth of such an environment, we consider that the 3D point-cloud obtained from its scan with a Total Station Leica TS60 is precise and dense enough to be considered as ground truth. The horizontal and vertical angular resolution of the scanning is set to 0.05 degrees. To scan the area, the Total Station is placed on three different locations, to have different viewpoints on the trees. Each scan takes 40 to 60 minutes. At the end, we obtain a single consistent point-cloud from the area, containing 5.5×10^6 points. The localization of the robot is provided by an RTK-GPS fused with an IMU in an Extended Kalman Filter. The position of the robot is measured in 6 different locations with the Total Station. These points are used to estimate the transformation of the RTK-GPS based frame to the Total Station frame. Because the estimate of this transform is not perfect, the error in the localization of the robot is not homogeneous. It is better in some areas of the map than in some others. We do not correct this error in this work, and use that instead to show results with different level of precision in the localization. For this experiment, the map is build with Octomap

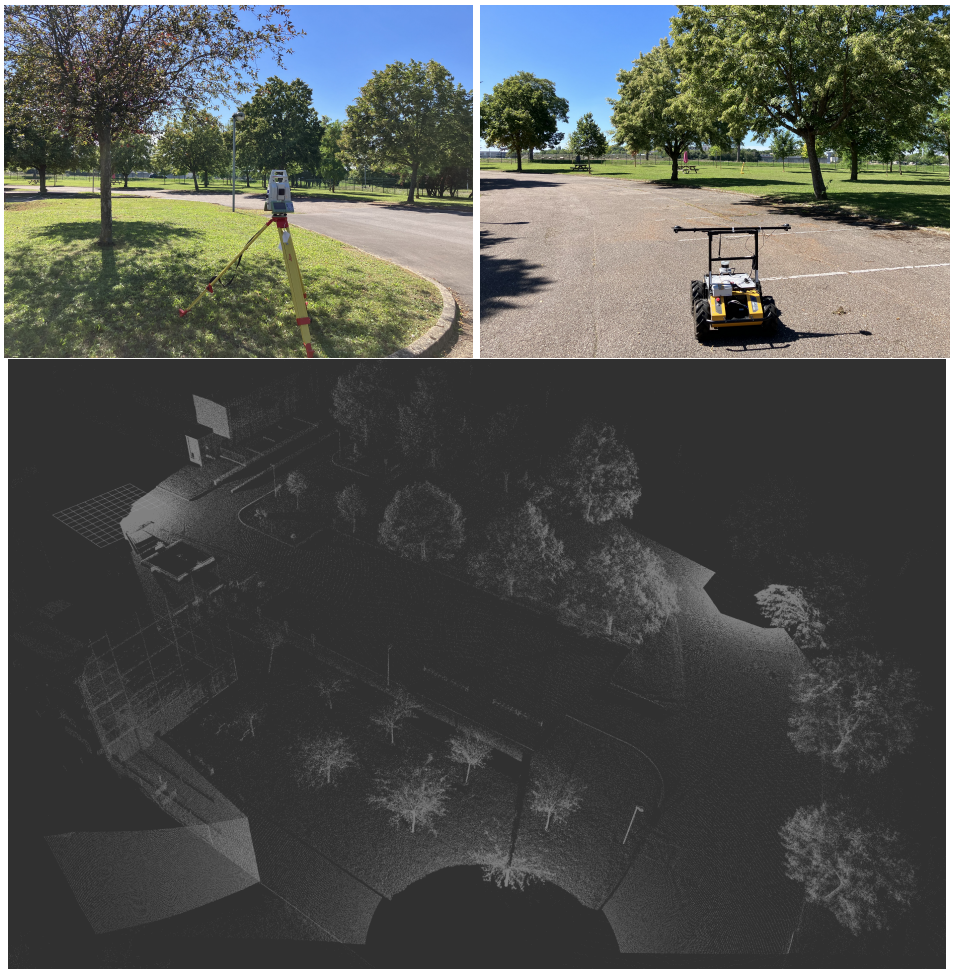

Figure 3.6: Illustration of a real world experiment. Top: right: the Leica Total Station, left: the Husky. Bottom: the point-cloud from the Leica

(max-range $10m$), with *res* $= 0.1m$, that we compare to the ground-truth, with a cuboid resolution of *RES* $= 1m$. Because of the small vertical field of view of the Lidar, the height of the Octomap is constrained to $6m$. Also, to avoid large errors due to large uncertainty in the orientation, we filter out the Lidar scans associated to turning motions of the robot when we build the map with Octomap.

3.2.3 Summary of the experiments

Type of worlds

- **RECT**: Experiment in a world with rectangular cuboids
- **CROSS**: Experiment in a world with cross extruded shapes
- **HELICOID**: Experiment in a world with helicoidal shapes

- **TREES**: Experiment in a world with simulated trees
- **REAL**: Real world experiment

Robot behaviors

- **NAVIGATION**: Experiments where the robot navigates autonomously towards a goal
- **TELEPORT**: Experiments where the robot is teleported on a goal
- **FRONT**: Experiment where the robot drives toward the object
- **LAT**: Experiment where the robot drives with the object on the side
- **WAYPOINTS**: Experiments where the robot follows autonomously a list of waypoints
- **RANDOM**: Experiments where the robot is teleported on a goal that is randomly sampled
- **NBV-XP**: Experiments where the robot is teleported on a goal selected by the policy

3.3 Summary

In this chapter, we presented firstly the method used for mapping and secondly our experimental framework. Regarding mapping, we presented our method for building a 3D-grid map from the robot's 3D-Lidar observations. This involved using Octomap and transforming it into a 3D-grid of occupancy likelihoods, along with converting ground-truth data into a 3D-grid. Additionally, we detailed our process for extracting intersecting cuboids from both the robot's map and reference map 3D-grids.

In the context of the experimental framework, we described our methodology for generating simulated environments, featuring different assets with varying complexities with respect to reconstruction. We also outlined various experiments that involved distinct robot behaviors, each designed to highlight specific aspects. Furthermore, we presented a real-world experiment in which ground-truth data is collected using a total station, ensuring consistency with our approach in simulation.

4

Measuring Map Quality

When the objective of an exploration policy is to enhance the map quality, such as in this work, evaluating the said policy requires first assessing the map quality. The motivation for the work detailed in this chapter arises from the challenges associated with evaluating 3D-map quality in natural environments. Typically, map quality evaluation aims to provide a single metric that reflects the overall map quality. Nonetheless, in the context we consider, it becomes interesting to obtain a localized measure of map quality. The map quality may not necessarily be homogeneous throughout a possibly large-scale environment. This is why we introduced the methodology presented in the previous chapter.

In the scenario considered here, a robot's 3D-lidar observations construct a 3D-grid map encoding the occupancy likelihood for each voxel. In this context, conventional map quality measures, like surface coverage and reconstruction accuracy, may not always hold significant meaning. This is specially true when dealing with natural environments that present specific challenges. As introduced in Chapter 2, the 3D-map in a natural environment is both unstructured and sparse, and the mapping is further complicated by the difficulties linked to the robot's localization within the map. We will demonstrate the limit of conventional metrics in this chapter, first in simulation, and then with a real world experiment. Nonetheless, it should be noted that this work does not directly evaluate various mapping algorithms applied to natural environments. Instead, our main emphasis lies in addressing the challenges associated with assessing the quality of 3D maps in such environments.

In a possibly large-scale environment, which could also be both sparse and unstructured, we try to answer the following questions: How can we evaluate the local map quality? Are the conventional metrics capable of delivering meaningful measurements? This study specifically centers around evaluating various 3D-reconstruction metrics, including both conventional and less conventional ones, with a specific focus on assessing their effectiveness in accurately measuring the map quality. First, we begin by selecting six relevant metrics: surface coverage, reconstruction accuracy, Average Hausdorff Distance, Cohen's Kappa coefficient, Kullback-Leibler Divergence, and Wasserstein Distance. Then, building upon the methodology presented in Chapter 3, we extract cuboid regions from both the reference map, called ground-truth, and the reconstructed map, called reconstruction, and by assessing the map quality with the previously mentioned metrics. Later on, we propose an additional methodology to evaluate the capability of the selected metrics in measuring various degradation models, when the reconstructed map is iteratively degraded from the reference map. Finally, we empirically compare the metrics in situations where the 3D map is built from point clouds obtained from the robot's observations, both in simulation and in a real world experiment. Ultimately, we present a comparison of the selected metrics, highlighting their properties, along with guidelines towards the choice of the metric depending on the application.

4.1 Related Work

4.1.1 3D reconstruction metrics

Classic 3D reconstruction metrics in robotics

In the context of 3D-map quality assessment, whether for robotics applications or not, the task typically consists in measuring the quality of the 3D reconstruction, often by comparing two surfaces. Commonly, those are the mesh generated from the 3D-point cloud, and the ground-truth mesh. Traditional metrics in that case are based on surface distance errors. They consist in calculating, for all surface points of one surface, the distance to the closest on the other surface, and then extract some statistics. Common surface distance errors are the **Hausdorff Distance**, that we will detail later, the **Root Mean Square Error** (RMSE) or the **Mean Average Error** (MAE) (used respectively in [6], [83], [84] for instance).

In the context of robotics, the prevailing metrics of reconstruction quality are the **surface coverage** and the **reconstruction accuracy**. Both are derived from those surface distance errors, and applied either on the meshes or on the 3D-grids. In the latter case, two sets of points are compared. Provided we want to compute the surface coverage, we are interested on the proportion of the set of points from the ground-truth accurately reconstructed. To do so, if the distance between a ground-truth point and its closest reconstructed point is less than a registration distance, the ground-truth point is considered as reconstructed (i.e valid). The metric is the proportion of such points ([36, 68]). Similarly, the reconstruction accuracy corresponds to the proportion of accurately reconstructed points in the set of points from the reconstruction, that is, points whose distance to the closest ground-truth point is below a registration distance.

Other reconstruction metrics

Some metrics have been proved efficient when it comes to evaluate the quality of the semantic segmentation predicted by deep-learning models, although most of them have been used outside the field of computer vision for decades. The idea with those metrics is to evaluate at the same time the classification accuracy and the correctness of the localization. Among them, we can cite the **accuracy**, the **precision**, the **recall**, or even the **Intersection Over Union** (IoU, Jaccard Index), or the **F1 score**. [72] provides a thorough review and comparison of the existing segmentation metrics for 3D-medical image segmentation tasks. Building upon [72], [53] compares some of those metrics depending on the size of the regions of interest to segment. As [72], building on the fact that we are working on 3D-grids, we can consider our problem a 3D-segmentation problem. We can then consider each voxel is assigned the class

"empty" or "occupied" based on its occupancy likelihood. In natural environments, the volume is mostly empty space, with sparse objects whose contours represent the occupied space. [53] shows that two metrics are particularly sensitive when it comes to measuring segmentation quality of small objects in an image: the **Average Hausdorff Distance** and **Cohen's Kappa coefficient**.

The Hausdorff Distance (HD) is, as the other metrics seen before, a spatial distance metric, widely used to evaluate 3D-reconstruction [6, 16]. HD is a common measure of distance between two point sets, but it is sensitive to outliers. [72] proposes to use instead the Average Hausdorff Distance (*AHD*), introduced in [63]. The *AHD* averages the HD over all the points, becoming more stable and less sensitive to outliers than HD.

The other interesting metric pointed out by [53], is the Cohen's Kappa coefficient (*KAP*). Unlike the metrics seen previously, *KAP* is not a spatial distance metric but a probabilistic metric. It was first proposed in [23]: it provides a score measuring the agreement between two samples. As an advantage over other measures, *KAP* takes into account the agreement caused by chance, which makes it more robust. That is, *KAP* is in [-1,1], where 1 corresponds to complete agreement, -1 to complete opposition, and 0 to random.

4.1.2 Comparing probabilities

Since we are building a probabilistic volumetric map with Octomap, we could take advantage of that framework to measure the quality of the map. Each voxel in the probabilistic map has a probability of occupancy between 0%, the absolute certainty that the voxel is empty, and 100%, the absolute certainty that the voxel is occupied. Leveraging the probabilities inside a 3D-grid is not a novelty, and has been explored in [36]. However, they do not propose a mean to compare the reconstructed volume to a reference one. They calculate the entropy of the voxels, which represents their distance to the unknown, thereby indicating the quantity of observation for each voxel, but not a measure of the reconstruction quality.

In this paper, we propose a methodology to compare two volumetric maps with probabilistic values. Different methods allow comparing probabilities. The most common is probably the **Kullback-Leibler Divergence** (*DKL*). The *DKL* [45] is a measure of how different a probability distribution is from another probability distribution. With the *DKL*, we can measure how the probability distribution of the reconstruction is different from the probability distribution of the ground-truth.

Nonetheless, since we are considering a grid of probabilistic voxels, we have access to another information: the Euclidean distance between voxels. As an example, if a point is erroneously reconstructed 5 cm away from an actual object, the reconstruction is better than

if the erroneous point is 50 cm away. The DKL is not sensitive to this difference.

An alternative solution is then to find the Optimal Transport plan, linking one probability distribution to another one, with a cost function depending on the geometry [74]. From this Optimal Transport plan, we can calculate a distance, the **Wasserstein Distance** (WD) which is a generalization of the concept of Earth Mover's Distance (EMD). Computing the Optimal Transport Plan may be cumbersome, because it is an optimization problem that is not necessarily convex. To bypass computational issues, [26] regularizes the optimal transport problem by adding an entropic term, and solves it using a Sinkhorn's fixed point iteration. [28] provides an open source Python library implementing several solvers for Optimal Transport problems, including [26]'s algorithm. With this regularized Wasserstein Distance, we can measure the quality of the 3D-reconstruction, comparing not only the ground-truth and reconstructed values of probabilistic maps but also taking into account the Euclidean distances in the errors.

4.2 Method

4.2.1 Comparison metrics

This section explains how, for each cuboid region, we compute different metrics to indicate the local map quality. Here, we call for convenience *reconstruction* the reconstructed 3D-map, and *ground-truth* the reference map. The cuboid extraction of reconstruction and ground-truth is described in Section 3.1.2. A cuboid from the reconstruction is denoted C_{rec}, and its corresponding cuboid from the ground-truth C_{gt}. All the code is available open-source[7].

Before proceeding, since this method is developed with the objective to work also in natural environments with sparse objects, the reconstructed volume may contain more empty space than actual objects to reconstruct. A measure that would give information on occupied space only may not be representative of the complete volume. For this reason, we found it interesting to measure not only how well objects have been reconstructed, but also how well the empty space has been reconstructed. To do so, we first define two sets: $\mathcal{U}_{occ}$, the set of the cuboids regions containing at least one occupied voxel in C_{gt} and $\mathcal{U}_{empty}$, its complement. Then, we measure the reconstruction quality of the cuboids with a different metric in each set: one of the considered reconstruction metrics for the cuboids in $\mathcal{U}_{occ}$, the L_1 norm for the cuboids in $\mathcal{U}_{empty}$. This study focuses on the evaluation of metrics in $\mathcal{U}_{occ}$.

Furthermore, because measuring reconstruction quality of unknown space is pointless, we consider a threshold before calculating our metrics. If all the probabilities in C_{rec} are

[7] https://github.com/stephanie-aravecchia/3d-reconstruction-metrics

close to the unknown (0.5 ± 0.1), we do not calculate the metrics, but set them to default values. Algorithm 1 summarizes this section.

Algorithm 1 Reconstruction and ground-truth comparison

// $\mathcal{D}_{gt}$ and $\mathcal{D}_{rec}$ are the datasets
$\mathcal{D}_{\text{gt}}(B_{\text{box}_1}, (h_1, w_1, n_1), R_f)$, $\mathcal{D}_{\text{rec}}(B_{\text{box}_2}, (h_2, w_2, n_2), R_f)$
$B_{\text{box}} \leftarrow B_{\text{box}_1} \cap B_{\text{box}_2}$
$M_{gt} \leftarrow \mathcal{D}_{\text{gt}}(B_{\text{box}}), M_{rec} \leftarrow \mathcal{D}_{\text{rec}}(B_{\text{box}})$
for all cuboid $\in B_{\text{box}}$ **do**:
    $C_{gt} \leftarrow M_{gt}$(cuboid), $C_{rec} \leftarrow M_{rec}$(cuboid)
    if isObserved(C_{rec}) **then**:
        if cuboid $\in \mathcal{U}_{occ}$ **then**:
            cuboid.metrics $\leftarrow$ computeMetrics(C_{rec}, C_{gt})
        else:
            cuboid.metrics $\leftarrow$ computeL1(C_{rec})
        end if
    else:
        cuboid.metrics $\leftarrow$ maxMetrics
    end if
end for

Choice of Metrics

As discussed in Section 4.1, several metrics are available to measure map quality, whereas they are 3D-reconstruction metrics, or measure of distance between probabilities. We select six of them for evaluation, based on the criteria listed in Table 4.1.

Acronym	Complete Name	Properties
COV	Surface Coverage	Surface distance-based score, widely used in robotics
ACC	Reconstruction Accuracy	Surface distance-based score, widely used in robotics
AHD	Average Hausdorff Distance	Surface distance, robust to small objects
KAP	Cohen's Kappa coefficient	Score, robust to small objects
DKL	Kullback-Leibler Divergence	Widely used for distance between probabilities
WD	Wasserstein Distance	Distance between probabilities (Euclidean distance with the cost matrix)

Table 4.1: Criteria for evaluated metric selection

Surface Distance Metrics

The three surface distance metrics we consider here (surface coverage, reconstruction accuracy and Average Hausdorff Distance) are based on spatial distances between sets of points. To compute them, we define the following constants:

- $\hat{p}$ is an occupancy likelihood threshold,
- d_r is a registration distance.

We also define the following variables:

- $\mathcal{U}_{rec}$ is the set of points P from C_{rec} whose occupancy likelihood is above $\hat{p}$,
- $\mathcal{U}_{gt}$ is the set of occupied points from C_{gt},
- n_{rec} is the number of points in $\mathcal{U}_{rec}$,
- n_{gt} is the number of points in $\mathcal{U}_{gt}$,
- $\|PS\|$ is the Euclidean Distance between a point P from $\mathcal{U}_{rec}$ to the closest S in $\mathcal{U}_{gt}$,
- $\|SP\|$ is the Euclidean Distance between a point S from $\mathcal{U}_{gt}$ to the closest P in $\mathcal{U}_{rec}$.

As an example, in Fig. 3.2, the points from $\mathcal{U}_{gt}$ and $\mathcal{U}_{rec}$ correspond respectively to the white voxels in C_{gt} and $\hat{C}_{rec}$.

Surface Coverage

To compute *COV*, we apply the classical methodology in a 3D-grid, as [36], to our cuboids. We first compute the number of reconstructed points, k_{rec}, that is, points in $\mathcal{U}_{gt}$ we consider correctly reconstructed, and then we compute the surface coverage, *COV*, the proportion of correctly reconstructed points.

$$k_{rec} = \sum_{S \in \mathcal{U}_{gt}} \mathbf{1}_B(\|SP\| \leq d_r) \tag{4.1}$$

where $\mathbf{1}_B(b) = 1$ if b, 0 otherwise.

$$COV = k_{rec}/n_{gt} \tag{4.2}$$

Reconstruction Accuracy

To compute *ACC*, we proceed similarly: we first compute the number of accurate points, k_{acc}, that is, points in $\mathcal{U}_{rec}$ we considered valid, and then we compute the reconstruction accuracy, *ACC*, the proportion of valid points.

$$k_{acc} = \sum_{P \in \mathcal{U}_{rec}} \mathbf{1}_B(\|PS\| \leq d_r) \tag{4.3}$$

$$ACC = k_{acc}/n_{rec} \tag{4.4}$$

Average Hausdorff Distance

To compute *AHD*, we follow [72]. The Hausdorff Distance measures the distance between two sets of points. We compute two Hausdorff Distances: the distance from reconstruction to ground-truth d_{PS}, and the distance from ground-truth to reconstruction d_{SP}:

$$d_{PS} = \frac{1}{n_p} \sum_{P \in \mathcal{U}_{rec}} \|PS\| \tag{4.5}$$

$$d_{SP} = \frac{1}{n_s} \sum_{S \in \mathcal{U}_{rec}} \|SP\| \tag{4.6}$$

Then, we compute the Average Hausdorff Distance, which consists in the maximum between the two distances:

$$AHD = max(d_{PS}, d_{SP}) \tag{4.7}$$

Cohen's Kappa

The metric Cohen's Kappa, *KAP*, provides a score in [-1, 1] ([worst, best]). This score provides a measure of agreement between two sets of classification. Unlike surface distance metrics seen before, we compare here directly the elements of C_{rec} and C_{gt}. In the cuboids, we consider our problem as a binary classification problem: each voxel is assigned either the class occupied or empty. In C_{gt}, each voxel already has a value 0 (class empty) or 1 (class occupied). In C_{rec}, we consider a voxel occupied if its occupancy likelihood $p > \hat{p}$, else, we consider it empty and call the resulting binary cuboid $\hat{C}_{rec}$. Fig. 3.2 provides an example. We iterate on all the voxels or of $\hat{C}_{rec}$, compare them to their corresponding voxel in C_{gt}, and we count *FP* (occurrence of false positive), *FN* (false negative), *TP* (true positive), *TN* (true negative).

Then, to compute *KAP*, we follow [72]. Let N be the number of voxels in a cuboid:

$$f_c = \frac{(TN + FN)(TN + FP) + (FP + TP)(FN + TP)}{N} \tag{4.8}$$

$$KAP = \frac{(TP + TN) - f_c}{N - f_c} \tag{4.9}$$

Kullback-Leibler Divergence

The DKL provides a measure of how a probability distribution is different from a reference probability distribution. The *DKL* metric we use in this work is the sum of the DKL between

the probability distributions derived from the elements of the cuboids. Let p^0 be the occupancy likelihood of a voxel in C_{rec}, g^0 the occupancy likelihood of the same voxel in C_{gt}. For numerical reasons, we saturate p^0 and g^0 in [m, 1-m], where m is a small number. The saturated values are p and g. Then, we iterate on the $N = n \times n \times n$ elements of the cuboids, and we compute the *DKL* metric as follows:

$$DKL = \sum_{k=1}^{k=N} \left[(1 - p_k) \cdot \log \frac{(1 - p_k)}{(1 - g_k)} + p_k \cdot \log \frac{p_k}{g_k} \right] \tag{4.10}$$

Wasserstein Distance

The Wasserstein Distance is derived from the optimal transport plan to "move" the mass distribution from a query vector to match the mass distribution of a reference vector. The cost of moving the mass being a function of the Euclidean distance it has to be moved by. Here, we calculate the Wasserstein Distance between two cuboid regions.

As this Wasserstein Distance is defined on histograms, that is, vectors that sum to 1, we first need to remap all the elements of C_{gt} and C_{rec} into two vectors of doubles, V_{gt} and V_{rec}, such that:

$$\forall (i, j, k) \in [0, n], V_*(i \cdot n^2 + j \cdot n + k) = C_*(i, j, k)$$

Second, we derive from each vector, two different vectors. From V_{gt}, we derive V_{gt}^{occ} and V_{gt}^{free}:

$$V_{gt}^{occ} = \max(2V_{gt} - 1, 0)$$

$$V_{gt}^{free} = \max(1 - 2V_{gt}, 0)$$

They contain respectively the probability of an element to correspond to an occupied voxel, and the probability of an element to correspond to an empty voxel. We then normalize each vector by their sum and obtain two histograms P_{gt}^{occ} and P_{gt}^{free}. Similarly, from V_{rec}, we obtain P_{rec}^{occ} and P_{rec}^{free}. We do this partition between occupancy and emptiness because we observed that the Wasserstein Distance between P_{rec}^{occ} and P_{gt}^{occ}, which embeds in each element the distance from its probability of occupancy to the unknown, contains more signal. Moreover, this corresponds better to what we intend to measure with this metric, that is how well the occupied space has been reconstructed. Using the same mapping between voxels and elements of the vector, we set the cost matrix M to contain the squared Euclidean distance between the voxels associated to the elements of the vector.

Finally, we calculate WD_{occ} the regularized Wasserstein Distance between P_{rec}^{occ} and P_{gt}^{occ}, computed using the Sinkhorn algorithm described in [26], following the implementation of [28]. This algorithm is an optimization that seeks an optimal coupling which minimizes the

displacement cost of a discrete measure, P_{rec}^{occ} in our case, to a discrete measure, P_{gt}^{occ}, with respect to a cost, a transport matrix, M, under an entropic constraint. The optimal value of the optimal transport problem is the Wasserstein Distance, defined as follows, with $\gamma\mathbf{1} = P_{rec}^{occ}$, $\gamma^T\mathbf{1} = P_{gt}^{occ}$ and $\gamma \geq 0$:

$$\textit{WD}_{occ} = \min_{\gamma} \left(\langle \gamma, M \rangle + \alpha \cdot \Omega(\gamma) \right) \tag{4.11}$$

Where M is the previously defined cost matrix, Ω is the entropic regularization term: $\Omega(\gamma) = \sum_{i,j} \gamma_{i,j} log(\gamma_{i,j})$, α is an entropic regularization factor and $\langle ., . \rangle$ is the Frobenius dot-product.

In this study, we set the regularization factor $\alpha = 1.0$. They show in [26] that the smaller the value of α, the better the precision of the algorithm, but also the slower the convergence. The value of 1 appears to be a good tradeoff, and produced satisfying results in our tests.

L_1 norm

In a cuboid region of the ground-truth in $\mathcal{U}_{empty}$, all the voxels have a probability of occupancy of 0. Therefore, when comparing the reconstruction to the ground-truth, we are comparing a vector containing some probabilities of occupancy V_{rec} to a vector of the same size containing only zeros (absolute certainty of emptiness). Such a measure is given by the L_1 norm of V_{rec}: the distance between V_{rec} and the vector of zeros that represent the empty space. We do not evaluate L_1 in this study, but simply provide it to the reader, because we found it convenient in other works:

$$L_1 = \sum_{(i,j,k)=(0,0,0)}^{(i,j,k)=(n,n,n)} C_{rec_{i,j,k}} \tag{4.12}$$

4.2.2 Evaluation Methodology

Controlled 3D-reconstruction

This section describes the evaluation of the metrics, with a specific focus on examining their behavior in the context of decreasing reconstruction quality. We present the methodology using a single ground-truth cuboid. The key approach involves initially measuring the reconstruction quality when it is perfect, indicated by $C_{rec} = C_{gt}$. Then, we introduce various degradation models that are iteratively applied to the reconstructed cuboid, progressively degrading the quality of the reconstruction. This methodology is inspired from biological approaches, like [11], where the evolutionary distance between a pair of gene sequences is usually measured by the number of edit operation (substitutions, insertions and deletions)

needed to transform one into the other. This distance is called the Levenshtein distance, or edit distance. Although this is not applicable here, our methodology is inspired from this concept: we apply a sequence of basic "edit" operations to increase the distance between reconstruction and ground-truth. Let us assume v is a voxel randomly sampled in the cuboid C_{rec}, $p(v)$ the occupancy likelihood of v, and v' a direct neighbor of v. The degradation models considered in this study are:

1. random occupied voxel $\mathcal{N}_{\text{occ}}$ $p(v) = 1$
2. random free voxel $\mathcal{N}_{\text{free}}$ $p(v) = 0$
3. random unknown voxel $\mathcal{N}_{\text{unknown}}$ $p(v) = 0.5$
4. random random voxel $\mathcal{N}_{\text{random}}$ $p(v) \sim U(0, 1)$
5. random shifted voxels $\mathcal{N}_{\text{shift}}$ $p(v') = p(v), p(v) = p(v')$
6. random flipped voxel $\mathcal{N}_{\text{flip}}$ $p(v) = 1 - p(v)$

Metric Evaluation and Comparison

We evaluate the different metrics with a consistent perspective, focusing on their ability to discriminate reliably "good" from "less good" reconstructions. Before going further, it is important to note that some metrics are distance metrics (the lower, the better): *DKL*, *AHD*, WD_{occ}, whereas the others are score metrics (the higher, the better): *COV*, *ACC*, *KAP*. First, following our inspiration of distances between sequences introduced before, we define the threshold $\hat{n}$ that divides the population of C_{rec} in two, the "good" ones, and the "less good" ones: $\hat{n}$ is simply a fixed level of degradation of the cuboid (i.e 20%). Second, we define the threshold $\hat{\theta}$ that divides the population of C_{rec} in two: the cuboids measured as "good" and those "less good". In the case of distance metrics, a measure θ smaller than $\hat{\theta}$ indicates a "good" measured reconstruction (conversely for score metrics). Finally, we consider our problem as a classification problem, and we populate a confusion matrix by evaluating all the cuboids, at all the iterations as explained in Tab.4.2. It should be noted that the definition of true / false positive / negative described here are different from the quantities used to compute Cohen's Kappa, where the classification was made on the voxels.

	score metric	*distance metric*
true positive	$(n \leq \hat{n})$ & $(\theta > \hat{\theta})$	$(n \leq \hat{n})$ & $(\theta \leq \hat{\theta})$
true negative	$(n > \hat{n})$ & $(\theta \leq \hat{\theta})$	$(n > \hat{n})$ & $(\theta > \hat{\theta})$
false positive	$(n > \hat{n})$ & $(\theta > \hat{\theta})$	$(n > \hat{n})$ & $(\theta \leq \hat{\theta})$
false negative	$(n \leq \hat{n})$ & $(\theta \leq \hat{\theta})$	$(n \leq \hat{n})$ & $(\theta > \hat{\theta})$

Table 4.2: Condition tested on a cuboid to populate the Confusion Matrix

From the confusion matrix, where we count the occurrence of true positive *TP*, true neg-

ative *TN*, false positive *FP*, and false negative *FN*, we compute the precision and recall of the metric:

$$precision = TP/(TP + FP) \tag{4.13}$$

$$recall = TP/(TP + FN) \tag{4.14}$$

We repeat the process for different thresholds $\hat{\theta}$ for each metric. From this data, we are able to compare the precision recall curves for the different metrics.

4.3 Experiments and Results

4.3.1 Experiments

The experiments are based on the framework described in Section 3.2.1. Firstly, we sample randomly 1500 cuboids from 12 simulated worlds containing rectangular cuboids, cross extruded shapes, helicoidal cones or simulated trees. We apply the controlled degradation described in Section 4.2.2 to these cuboids.

Secondly, we build 3D-reconstructions from the robot's observations in the same environments from the RANDOM experiments, where the robot is teleported randomly 100 times. For each of the 12 environments, for each of the 3 noise levels, we run 28 simulations, for a total of 1008 experiments in simulation.

Finally, in worlds containing only a single asset of each type, we run the two straight lines experiments: FRONT and LAT.

4.3.2 Results

In this section, we assess the metrics considered in this study for their capacity to offer a meaningful quality measure. We present a selection of real-world experiment cuboids, illustrating the challenge of assigning a quality score to maps, even for human observers. Thus, these examples offer insights into metric behavior, but rigorous statistical validation requires controlled maps to compare to reference maps. Consequently, the evaluation is run in simulation, where the reconstruction comes from an iterative degradation of the ground-truth, and the real-world experiments validate the evaluation.

Before delving into details, we would like to offer some preliminary information to help the understanding of this section. Firstly, Table 4.3 provides the acronyms used to refer to the simulated environments and the degradation models.

Secondly, Table 4.4 summarizes the classification of the metrics: for score metrics, higher is better, whereas for distance metrics, lower is better.

Acronym	**Type of asset in the world**
RECT	Rectangular cuboids
CROSS	Cross extruded shape
HELICOID	Helicoidal cone
TREE	Simulated tree
	Type of experiment
RANDOM	The robot is teleported 100 times randomly
FRONT	The robot drives toward the object
LAT	The robot drives with the object on the side
	Degradation model
$\mathcal{N}_{\text{occ}}$	a random voxel is set to occupied
$\mathcal{N}_{\text{free}}$	a random voxel is set to free
$\mathcal{N}_{\text{unknown}}$	a random voxel is set to unknown
$\mathcal{N}_{\text{random}}$	a random voxel is set to random
$\mathcal{N}_{\text{shift}}$	the occupancy likelihoods of two random neighbor voxels are shifted
$\mathcal{N}_{\text{flip}}$	the occupancy likelihood of a random voxel is set to its emptiness likelihood

Table 4.3: Summary of the acronyms used in the Results section

Score Metrics	Distance Metrics
Surface Coverage *COV*	Average Hausdorff Distance *AHD*
Reconstruction Accuracy *ACC*	Kullback-Leibler Divergence *DKL*
Cohen's Kappa *KAP*	Wasserstein Distance WD_{occ}

Table 4.4: Classification of the metrics

Finally, as detailed in Sec. 4.2.1, surface distance metrics rely on constants, namely occupancy likelihood threshold and registration distance. For our experiments, we used common values in robotics applications (similar to [36, 68]):

- $\hat{p}$: 0.7, **0.8** (*COV*, *ACC*, *KAP*, *AHD*),
- d_r: **0.05**, 0.1, 0.15 meters (*COV*, *ACC*).

For the sake of brevity, we present results only for the values in bold. Even though the metric value is slightly affected by the constants, the overall behavior remains consistent.

4.3.3 Theoretical Metrics Comparison

We compare the metrics when the reconstruction moves further from the ground-truth, following the methodology in Sec. 4.2.2, and we distinguish the experiments by type of world.

Figure 4.1 provides a visualization of the degradation models we are considering. The figure displays a single slice of a cuboid, from the ground-truth, and from the differently

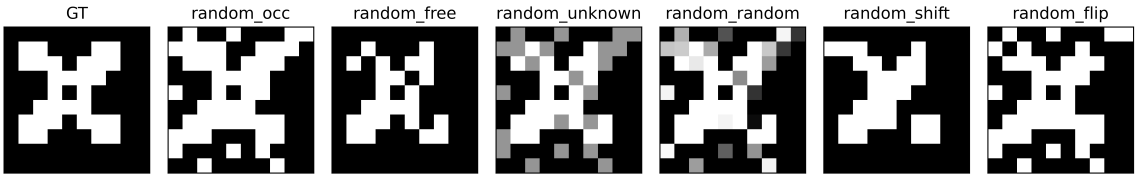

Figure 4.1: Visualization of the degradation models. Left-most is one slice of ground-truth cuboid. Others: degraded cuboids, after 200 iterations, with the different degradation models.

degraded cuboids, after 200 iterations.

Metrics behavior when C_{rec} moves further from C_{gt}

We now evaluate the metrics on the 1500 sampled cuboids detailed in Sec. 3.2.2, with the six different degradation models. A "good" metric is expected to: be sensitive to all the types of transformations, vary monotonically when the reconstruction moves further from the ground-truth, be independent of the type of world, and give measurements in a range that does not depend too drastically on the transformations. Under this assumption, we plot for each metric, for each type of degradation, and for each type of world, how the metric behaves when the reconstruction moves further from the ground-truth. Fig. 4.2 shows the results in the CROSS worlds, and the complete graph is provided in Appendix A. The conclusion from these graphs is that no metric satisfies all those conditions.

From each individual graph in Fig. 4.2 corresponding to a couple (metric, type of degradation), we can distinguish three trends, depending on how the metric vary when the reconstruction moves further from the ground-truth (n increases).

- no variation, the curve is flat: the metric is not sensitive to the type of degradation;
- a small variation (gentle slope): the metric seems slightly sensitive to the type of degradation, but we cannot conclude;
- a huge variation: the metric is sensitive to the type of degradation.

Table 4.5 summarizes the results of all these graphs and displays the sensitivities of the metrics to the different type of degradation.

Another interesting conclusion from our evaluation is that the metrics are generally dependent on the type of world. We illustrate that statement with the two graphs in the bottom part of Figure 4.2. It shows the influence of the type of world for two metrics, *ACC* and *AHD*, under $\mathcal{N}_{\text{occ}}$ degradation model.The areas corresponding to 80% of the measures in TREES environments (red) are larger and the medians are different from the areas corresponding to

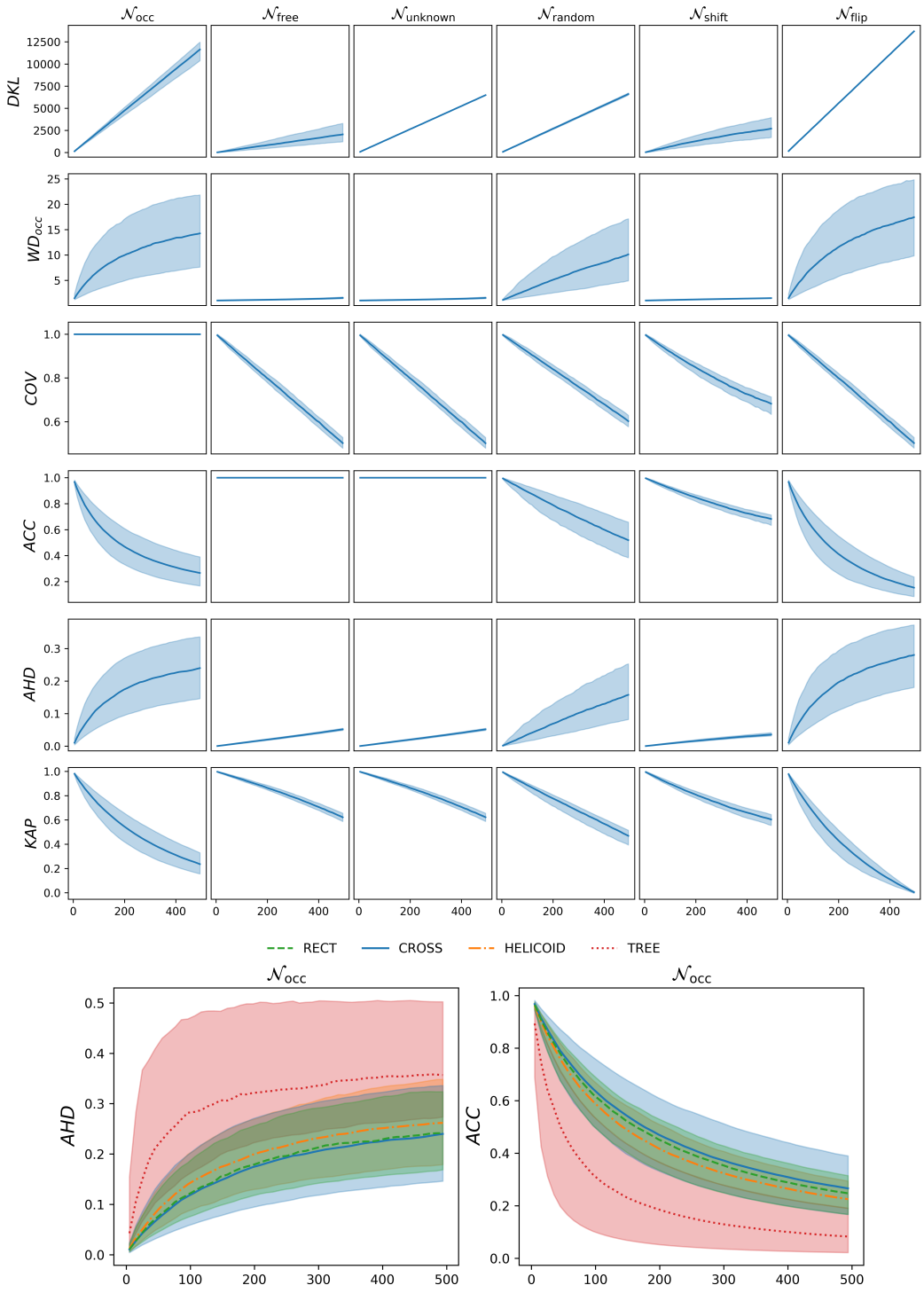

Figure 4.2: Top: illustration of the different metrics behavior when the reconstruction moves further from the ground-truth. One line per metric, one column per type of degradation applied to the gt. For each sub-figure, the x-axis is the number of iteration n, the y-axis the value of the metric θ. The results are displayed only for the CROSS worlds. In each sub-figure, the line corresponds to the median value from all cuboids and the filled area shows the spread of 80% of the population. Down: Same information, displayed in all the type of worlds for *AHD* and *ACC* with a $\mathcal{N}_{\text{occ}}$degradation model.

	$\mathcal{N}_{\text{occ}}$	$\mathcal{N}_{\text{free}}$	$\mathcal{N}_{\text{unknown}}$	$\mathcal{N}_{\text{random}}$	$\mathcal{N}_{\text{shift}}$	$\mathcal{N}_{\text{flip}}$
DKL	X	?	X	X	?	X
WD_{occ}	X	-	-	X	-	X
COV	-	X	X	X	X	X
ACC	X	-	-	X	X	X
AHD	X	?	?	X	?	X
KAP	X	?	?	X	?	X

Table 4.5: Metrics apparent sensitivity to the degradation types. -: not sensitive, x: sensitive, ?: unconclusive.

80% of RECT environments for instance (blue). That tends to indicate that not only the measure provided by the metrics are noisier in challenging environments, but also the very value supposedly dependent only on the reconstruction quality also depends highly on the type of environment. Again, the complete results are shown in Appendix A.

In real world applications, the noise model is probably a combination of all the degradation models considered here, and a central question is: is there a threshold to discriminate reliably “good" and “less good" reconstructions for the chosen metric ?

Precision-Recall of the metrics

Fig. 4.3 shows the very challenge of setting thresholds to discriminate “good" and “less good" reconstructions, with precision-recall curves. Precision-recall curves show the tradeoff between precision and recall for different thresholds. The better the classifier, the closer the precision to 1 for all values of recall. This figure shows the precision-recall curves obtained as detailed in Sec. 4.2.2, where the threshold $\hat{\theta}$ for each metric varies in a specific range. This range matches the min and max values of the y-axis of Fig.4.2 for the respective metric.

For each metric, the curves are drawn with 10 values of $\hat{\theta}$, and we highlight three particular values with the circle, diamond and star markers. Fig. 4.3 displays only results in the CROSS words. The complete graph is provided in Appendix A. Fig. 4.3 shows that, with a specific threshold (one of the markers), a metric can perform well for certain degradation models while performing poorly for others. Finally, the two plots in the right part of the figure show that the metrics' performance depends on the type of world: their lower performance is in the TREES worlds. Complementary to the questions marks in Table 4.5, these graphs suggest that *AHD* might be sensitive to $\mathcal{N}_{\text{free}}$ $\mathcal{N}_{\text{unknown}}$ and $\mathcal{N}_{\text{shift}}$, as might *KAP* in a slighter way. On the contrary, *DKL* might not be sensitive to $\mathcal{N}_{\text{free}}$ and $\mathcal{N}_{\text{shift}}$.

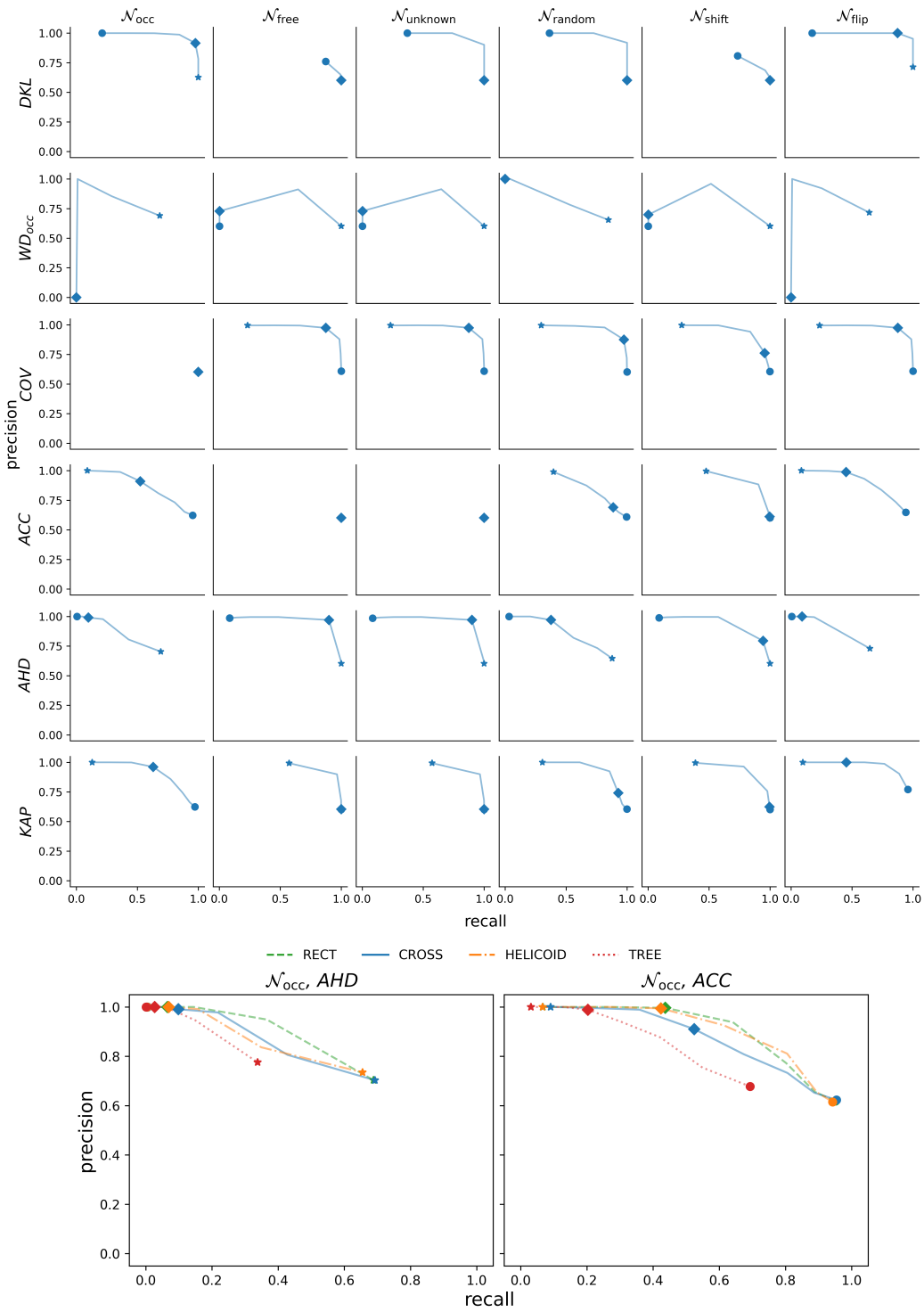

Figure 4.3: Left: Precision-Recall curves when we vary the value of the threshold $\hat{\theta}$ for each metric. One line per metric, one column per type of degradation. The results are shown only for CROSS worlds. The three markers display precision-recall points for three values of $\hat{\theta}$ for each metric. The results are displayed for a level of degradation of the cuboid of 20%. The points in (0,0) corresponds to points where it is not possible to compute precision and recall (division by 0 in Eq. 4.13 and 4.14) Right: Same information, displayed in all the type of worlds for *AHD* and *ACC* with a $\mathcal{N}_{\text{occ}}$ degradation model.

4.3.4 Metrics Comparison on experimental reconstructions

In this section, the comparison no longer focuses on the metrics' behaviors with controlled reconstructions, but is instead directed towards their performance when the map is built from the robot's observations.

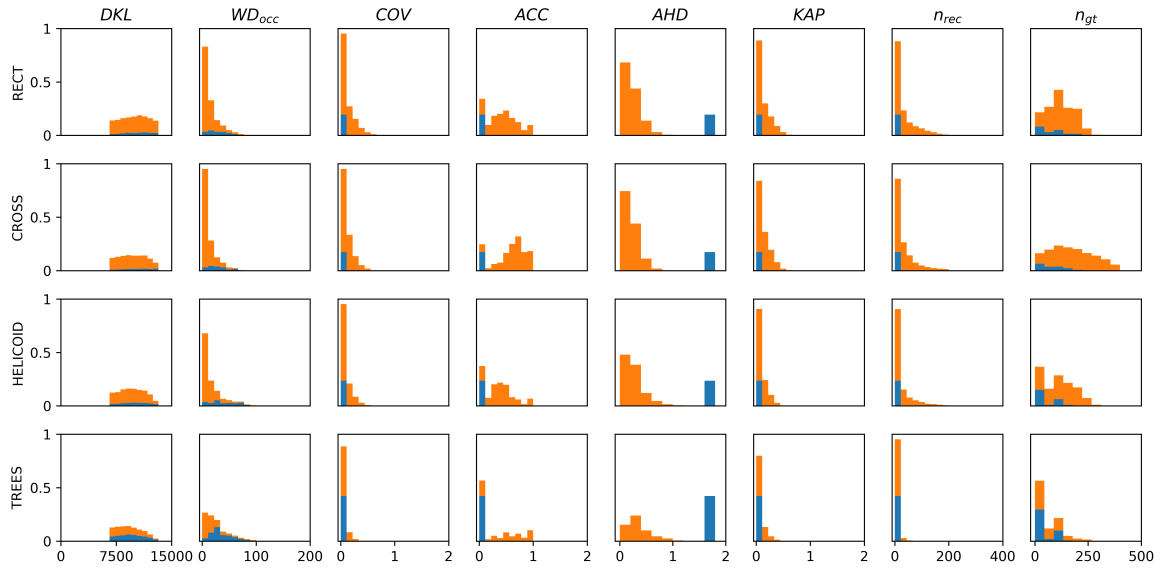

Figure 4.4: Distribution of the values of the metrics among the cuboids from $\mathcal{U}_{occ}$ in the experiments. The blue corresponds to the values of the cuboids where $n_{rec} = 0$. The orange to the other cuboids. The values are computed only for "observed" cuboids (at least one voxel in C_{rec} has $p < 0.4$ or $p > 0.6$, explained in Sec. 4.2.2). The figure display one line per type of world and one column per metric, with two extra-columns showing n_{rec} and n_{gt}.

Comparison of the metrics distributions

Fig. 4.4 shows the distribution of the different metrics per type of world. We consider only the cuboids in $\mathcal{U}_{occ}$, and we display also the distribution of n_{gt} and n_{rec}, the occurrence of occupied voxels in the ground-truth and reconstruction cuboids C_{gt} and C_{rec} (Sec. 4.2).

The graphs in Fig. 4.4 are arranged from top to down in increasing level of difficulty in the reconstruction, from RECT (basic geometrical shape), to TREES (unstructured objects). This figure shows that determining a meaningful threshold above or below which the reconstruction can be considered good is not straightforward. This is highlighted particularly with the blue color in the histograms, corresponding to cuboids where $n_{rec} = 0$ (denoted C^0_{rec}). These cuboids, where not a single point have been reconstructed, are likely "bad" cuboids.

Fig. 4.4 shows two trends:

- The metric is likely to provide a noisy measurement: the cuboids of C^0_{rec} are spread on all the range of values.
- The range of the values the metric provides shrinks when the complexity of the world increases.

From those two trends, we can hypothesize that *DKL* and WD_{occ} are likely to be noisy. We can also hypothesize that *AHD* is the most capable of providing measures when the difficulty in the world increases: the proportion of cuboids where the value is not in the first bar is the largest. Additionally, *AHD* provides measures only when there is at least a reconstructed point, making it easier to identify C^0_{rec} cuboids.

Table 4.6 summarizes the apparent advantages of the metrics. We hypothesize a metric may have this potential property if it does not follow the corresponding trends described above. Table 4.6, Table 4.5 and Figure 4.3 all indicate that one metric seems more potent in

Potential feature	*DKL*	WD_{occ}	*COV*	*ACC*	*AHD*	*KAP*
Limited noise in measurement	-	-	x	x	x	x
Robust to world complexity	x	x	-	-	x	-

Table 4.6: Apparent advantages of the different metrics

challenging environments: *AHD*.

Insight on potential metrics combinations

Complementary to Table 4.5 on controlled reconstructions, Figure 4.5 shows the correlation matrix of the different metrics computed on the cuboids containing at least one reconstructed point, in all the experiments. We computed this same correlation matrix by type of world, and by noise level in the localization, and although the level of correlation may change, the trend remains the same.

From that matrix, we can see that *DKL* is the metric the least correlated with the others. Our hypothesis is that *DKL* is sensitive to different information (we believe in the level of unknown in the cuboid) and may be combined with other metrics for a more reliable estimate of the reconstruction quality. Such combination is not straightforward. It would require normalizing the combined metrics in a range in which they all are meaningful, and add weighting factors. We leave that to future work.

Lastly, Table 4.7 presents the computation time associated with each metric. It is important to highlight that no specific optimization effort were applied to any of the implementations. Moreover, the computation of WD_{occ} could greatly benefit from running on GPU. Such

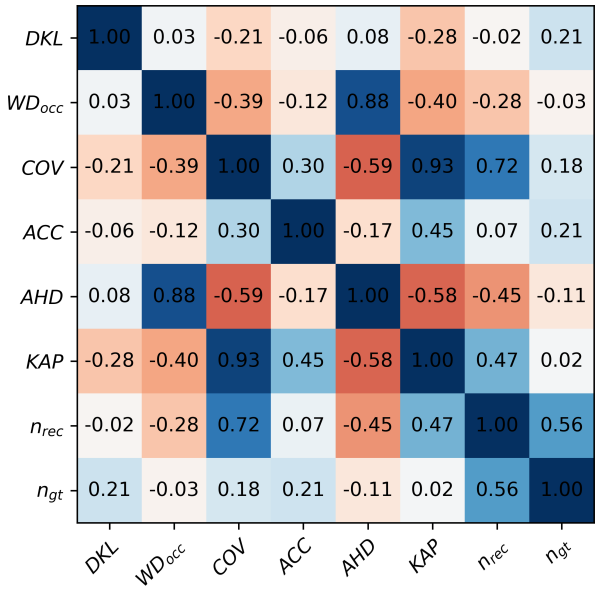

Figure 4.5: Correlation Matrix of the metrics. The correlation can be positive or negative depending on the type of metric (score or distance).

an implementation is provided by python-optimal-transport [8]. While not employed in this work due to its current C++ implementation and the absence of real-time inference requirements, this option holds potential. An interesting observation from this table is the efficiency of *DKL* computation, suggesting that combining it with another metric would demand a relatively small computational effort. Additionally, it's worth mentioning that *ACC* is faster to compute than *COV* because there are generally fewer points in the reconstruction than in the ground-truth. Finally, for those interested in the computation time, *KAP* stands as promising option.

	time (μ s)	$\pm$ std (μ s)
DKL	12	$\pm$ 1
WD_{occ}	137×10^3	$\pm$ 63×10^3
COV	230	$\pm$ 53
ACC	11	$\pm$ 17
AHD	241	$\pm$ 57
KAP	5	$\pm$ 1

Table 4.7: Comparison of the computation time

[8] `https://pythonot.github.io/index.html`

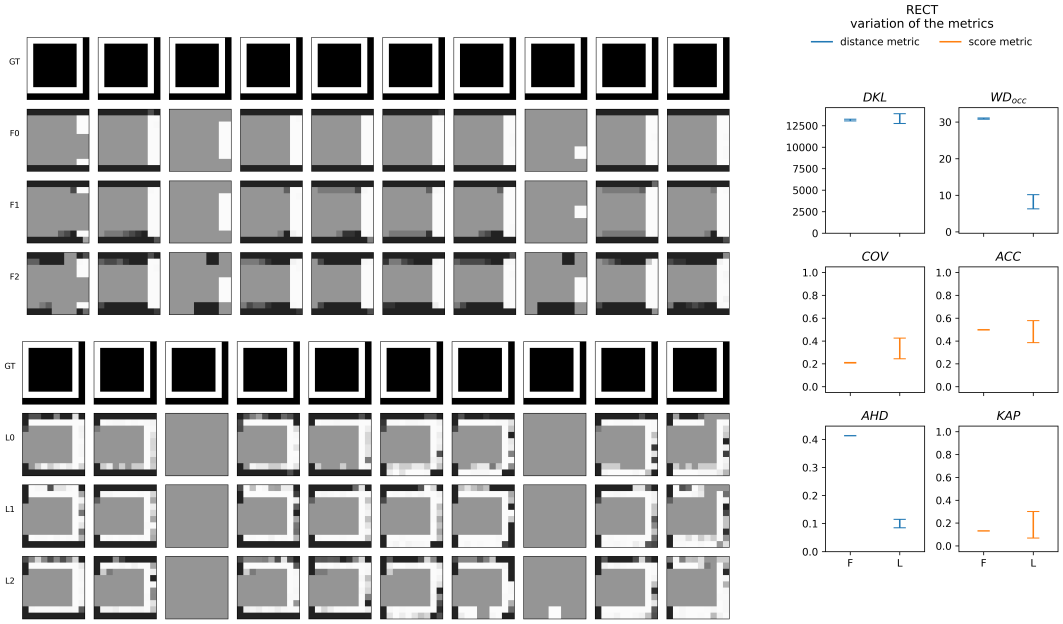

Figure 4.6: Example of a cuboid in a RECT environment. In the left, the first group of rows corresponds to the cuboids reconstructed with the FRONT driving behavior. GT is the ground-truth cuboid, F0, F1, F2 correspond to the reconstructions obtained with three noise level in the robot localization. The second group of rows corresponds to the LAT driving behavior. L0, L1, L2 to the reconstructions with the three noise levels. Two slices of the cuboids (3 and 8) remain mostly unknown: the Lidar used in this study is a 16-plane Lidar, and those slices remain situated between two of those planes during the experiments. The right part of the figure displays the values of the metrics, with the two different driving behaviors. Distance and score metrics are displayed with different colors to facilitate the interpretation (distance: lower is better, score: higher is better). The errorbars display the variation between the min and the max of each metric, for each driving behavior, when the reconstruction is built with the three noise levels in the robot localization.

4.3.5 Qualitative comparaison

Comparing reconstructions of the same cuboid

In this section, we intend to compare qualitatively the metrics on different reconstructions of the same ground-truth cuboid. To enable this comparison, we refer specifically to two sets of experiments in simulation, introduced in Section 3.2.1, the FRONT and LAT experiments. These experiments are designed to create two reconstructions that can be objectively ranked in terms of quality. We have deliberately selected our assets to emphasize the effects of occlusion. When we approach the object from the front (FRONT), a larger part of the object remains hidden compared to when we approach it from the side (LAT). As a result, the difference in reconstruction quality should be noticeable.

We then build the reconstructions from the robot's observations, with different noise level in the localization (as detailed in Sec.3.2.1).

Fig. 4.6 and Fig. 4.7 shows the results for the same cuboid, in the RECT and TREES environments, and the three reconstructions built from the two respective driving behaviors. We focus first on Fig. 4.6. In the context of an autonomous robot building a map, based on our expectations, we can presume that the maps from LAT are better than the maps from FRONT. Indeed, the group of reconstructions at the bottom of the figure appear "better" than the group at the top. We then expect the metrics to measure better reconstructions in LAT compared to FRONT. Nonetheless, the metrics yield divergent results when assessing the reconstruction quality, as they are not equally sensitive to all types of errors. For instance, *COV* measures a better reconstruction in LAT compared to FRONT, as a larger portion of the object has been reconstructed. Conversely, *ACC* measures a better map in FRONT compared to LAT, as the few points in FRONT are reconstructed more accurately. *KAP* is highly sensitive to noise in the localization: it penalizes erroneous points, regardless of the Euclidean error distance. Due to errors in the localization, discretization errors, or aliasing, an offset of one voxel, or pixel, is very likely, and a metric that penalizes such errors is very strict. We can see such errors in Fig.4.6. As a consequence, the variation of *KAP*'s measures in LAT includes the range of values of those in FRONT. In other words, with *KAP*, one LAT reconstruction is measured as better than the all FRONT reconstructions, whereas another one is measured as worse. On the contrary, *AHD* and WD_{occ} are robust to that type of errors, as they are to noise in the localization. They provide a measure that is consistent with what one would expect in the context of autonomous robot mapping, that is, LAT are better than FRONT. *DKL* is dominated by the unknown volume of the map. It does not appear useful in this example, but it might provide information complementary to the other metrics, for instance in an exploration task, where reducing the unknown is a central feature.

Fig. 4.7 shows the results in the most challenging simulated environment: TREES. First, when we analyze this figure, we cannot reach an obvious conclusion, as with the previous example. Visually, the reconstructions from LAT experiments do not appear significantly better compared to FRONT experiments. However, this trend is still what is indicated by all the metrics but WD_{occ}. This illustrates our claim that measuring 3D-reconstruction quality in such an environment is a challenging task in itself.

Also, we can point out that the metrics are generally sensitive to the density of points, either in the ground-truth or in the reconstruction. When one or both is really low, we reach the limit of all those metrics.

Comparing real-world cuboids

In this section, we focus on comparing the metrics when measuring quality of real-world cuboids. We select 8 cuboids, displayed in Figure 4.8. Cuboids A, B and C are correctly reconstructed. The reconstruction is noisy, but we can overall recover the underlying shape of the ground-truth, in the correct location. Cuboids D, E, F, G and H are poorly reconstructed. D, E, F are reconstructed with an error in the localization: the z error (represented by the offset in the sliced images) is visible. Apart from that, we can mostly recover the underlying shape of the ground-truth in the reconstruction. G and H are also reconstructed with a z error, but are overall difficult to "grade", because of the unstructured nature of the objects they contain (namely, branches). For a human observer, apart from assigning a better grade on cuboids A, B and C, grading all the poor reconstructions is a subjective task. The figures in the bottom of Figure 4.8 show how all the metrics measure these reconstructions quality. Firstly, most metrics (apart from *DKL*) generally agree that A, B, C (the first group of three symbols) are ranked in the best reconstructions. Secondly, two interesting observations emerge from the other cuboids. The first observation is that some metrics rank some poor cuboids as good as the good ones (F, G for WD_{occ}, E for *COV*, G, H for *ACC*). The second observation is that some metrics do not provide any information at all on those reconstructions (D, F for *COV*, *ACC* and *KAP*). The fact that the underlying structure of the ground-truth is present, even though with an error in the localization, is completely lost in the measure. From these figures, it seems the most robust metric is *AHD*, comforting the observations from simulation. Finally, those figures also show that combining metrics, for example *AHD* and *DKL*, would likely result in a more robust metric. For instance, by doing so, the unknown remaining volume in G or E would penalize their measured distance, something *AHD* alone cannot measure.

4.4 Summary

	DKL	WD_{occ}	*COV*	*ACC*	*AHD*	*KAP*
Sensitive to additional points	2	2	0	2	2	2
Sensitive to missing points	1	0	2	0	1	2
Informative wrt unknown volume	2	0	0	0	0	0
Fast to compute	2	0	1	2	1	2
Proportional to Euclidean distance error	0	2	2	2	2	0
Robust to noise	1	2	1	1	2	0
Robust to point density	0	0	0	0	1	0
Already normalized	0	0	1	1	0	1

Table 4.8: Summary of the metric properties. 0: the metric does not have this property, 1: it has it, but it is not a significant feature, 2: the property is significant

This chapter presented our methodology to assess map quality at a local level. Considering the cuboids regions introduced in Chapter 3, we computed six distinct metrics, namely, surface coverage (*COV*), reconstruction accuracy (*ACC*), Average Hausdorff Distance (*AHD*), Cohen's Kappa coefficient (*KAP*), Kullback-Leibler Divergence (*DKL*), and Wasserstein Distance (*DKL*), to assess locally the map quality. We developed a dedicated methodology to evaluate these metrics. We performed controlled 3D reconstructions by iteratively degrading ground truth cuboids using six different degradation models. Additionally, we proposed a methodology to assess the metrics' performance as two-class classifiers, where each class corresponds to a level of quality of the cuboids.

To summarize our results, we have seen that no metric is able to provide a meaningful measure in all the situations. Mainly, it depends on the intent of the quality measurement. Table 4.8 shows a summary of the metrics properties. In the context of autonomous robot mapping in natural environment, we believe a key feature is noise-robustness. When the environment is unstructured, *AHD* is probably a good choice of metric. *AHD* may be combined to *DKL* in the context of autonomous exploration, where it would also be interesting to have information on the remaining unknown volume. When the environment is structured, *ACC* or *COV* might still be good choices too, if the intent of the measure is to assess either the proportion of the ground-truth points reconstructed (*COV*), or the proportion of accurately reconstructed points (*ACC*), but not both at the same time. In a structured environment, their lack of robustness to point-density is counterbalanced with the underlying density of the ground-truth. Nonetheless, if the aim of the metric is to provide a robust measure of the reconstruction quality, then *AHD* remains a good choice, both in structured and unstructured environments.

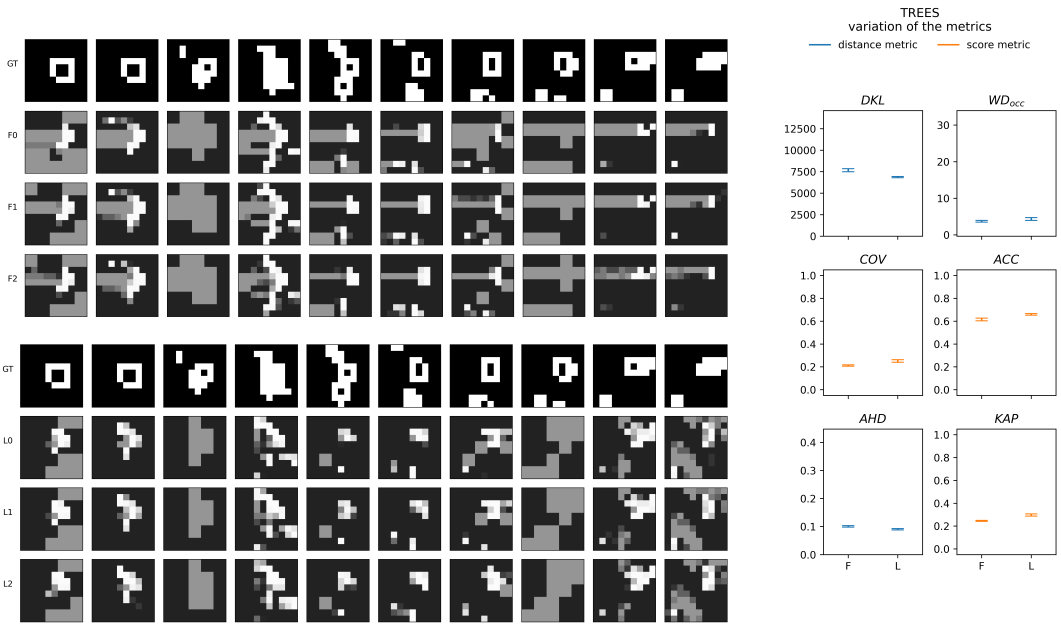

Figure 4.7: Example of a cuboid in a TREE environment. The figure display the same information as Fig. 4.6.

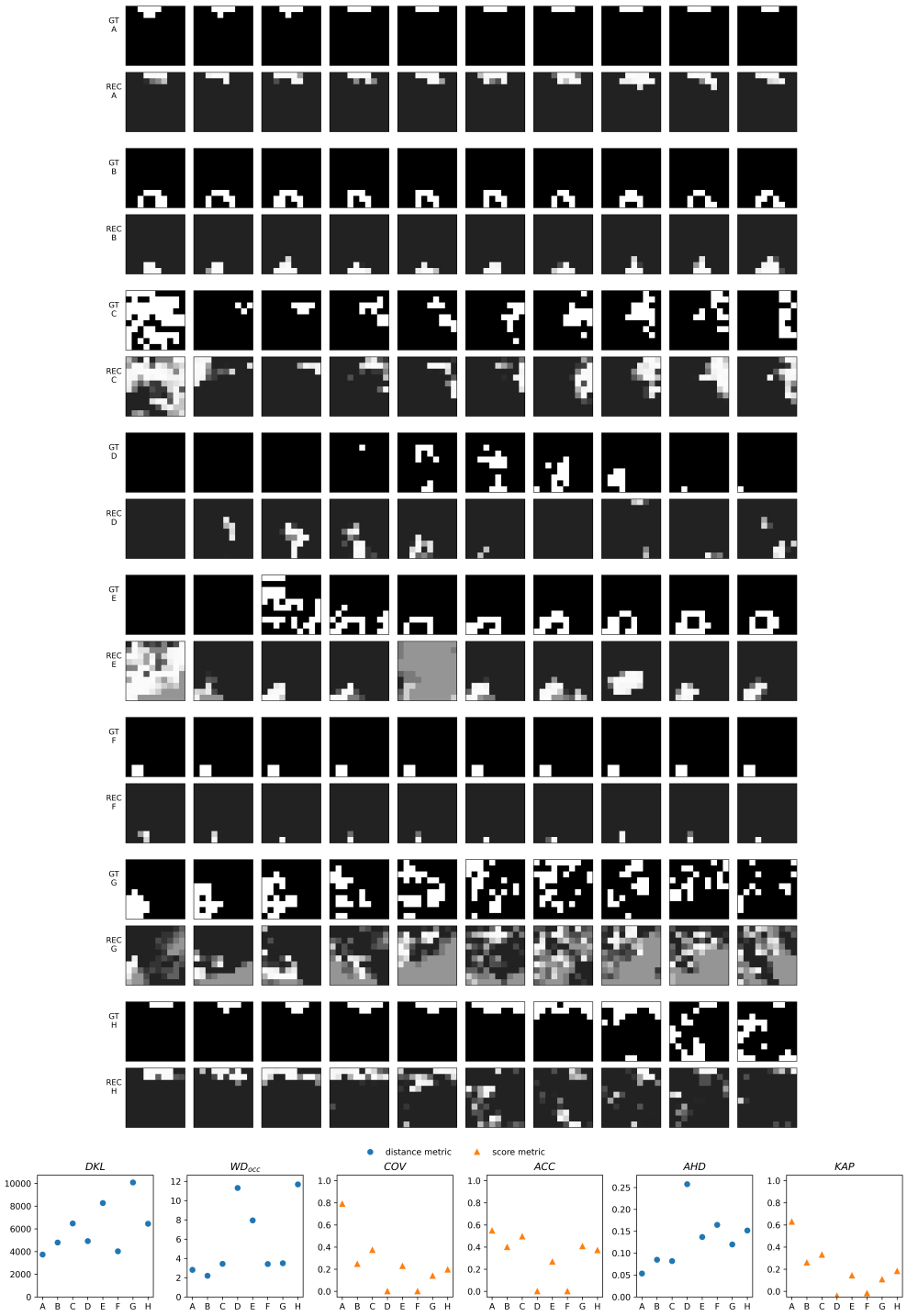

Figure 4.8: Example of a cuboid in a real environment. 8 cuboids are displayed (A to H). Each time, top row: C_{gt}, bottom row: C_{rec}. The last row shows the metrics corresponding to the 8 cuboids, one plot per metric. Distance metrics are blue circles, score metrics orange triangles.

5

Linking Map Quality and View-Point Statistics

When the primary goal of the exploration is to construct an accurate map, two central questions emerge. The first one is “how accurate is the map currently, in each section that is currently mapped?". However, in exploration missions, ground-truth data is typically unavailable. This leads to the second question: “how can we derive an exploration policy to enhance map quality, without access to a reference to compare the map against?".

This chapter presents a key contribution of this work, trying to answer these questions. Given the practical challenges of directly measuring map quality during exploration, we propose an alternative approach. We suggest instead estimating map quality a priori, relying on viewpoint statistics inferred directly from the robot's data. This estimation is performed at a local level, on the cuboids regions depicted in Chapter 3. By doing so, we can identify areas that are worth re-observing and how to re-observe them. Conversely, this approach allows us discarding areas that do not significantly enhance map quality. These two features are essential to build an effective exploration policy, as we will show in Chapter 6.

In this chapter, we first introduce the four computationally light viewpoint statistics that we study. The first two are conventional: the number of times a cuboid region is observed and the minimum range of the observations. The other two statistics are more original and stem from the particular characteristics of environments we consider, which are sparse and unstructured. These statistics are selected for their ability to express the diversity of the viewpoints: the number of angular sectors from which a cuboid is observed and the spherical variance of the viewpoints. Then, we provide a methodology to prove statistically that these viewpoint statistics are indicators of local map quality. Later on, we train a Random Forest Regressor to predict cuboid quality from viewpoint statistics. Finally, we measure the importance of each of these statistics on this prediction, using the permutation importance of each input feature.

5.1 Related Work

5.1.1 Factors impacting the map quality

Probabilistic Update

In 3D-grids, as in all probabilistic maps, the current map is actually a state. This state cannot be known with absolute certainty, but is instead estimated, introducing some level of uncertainty. The current map is a state estimation, derived from the history of the robot's observations. As introduced in Chapter 2, the robot's pose, the map, are beliefs. Their current state is an estimation build from a history of previous measurements and actions. Such recursive state estimation is generally based upon Bayes filters, as extensively detailed in [29]. We introduced occupancy grid maps in Chapter 2, and we are now providing more details,

following [29].

Occupancy grid maps address the problem of generating consistent maps from noisy and uncertain measurement data, from a known robot pose. The 2D occupancy grid was first introduced by [52]. The basic idea is to represent each cell of the map with a binary random variable, that corresponds to the occupancy of the location it covers. Each cell encodes its probability of being occupied, and its probability of being free. In occupancy grids, the state is considered static. Moreover, it is not estimated over the entire map, but the static state of each cell is estimated independently. The idea is that different measurements implying that a cell may be occupied should reinforce each other, while measurements that the cell is empty should weaken the certainty of it being occupied, and vice versa. As such, the more a cell in the map is observed, the lower its uncertainty. Although initially developed by [52] for sonars, which are particularly noisy sensors, this probabilistic update is the basis of several probabilistic mapping algorithm using Lidars. Among them, we can mention the widely used gmapping algorithm [32], or Octomap [35], the mapping framework used in this work.

From this update, it is evident that the uncertainty in a cell is directly linked to the number of times it is observed. From this, we can hypothesize that the quality of an area of the map is linked to the number of times it is observed. We will demonstrate this in this chapter.

Observational factors

Other observational factors may impact the map quality. Firstly, in unstructured environments, occlusions are frequent, due to a tree in front of another tree, a branch partially hiding a tree behind, and so on. One solution to circumvent occlusions is to observe from different viewpoints. Secondly, the map quality is directly linked to the sensor modality. In [69], they show that the quality of a point-cloud from a Lidar depends on the distance and the orientation of the scanned surface. The noise in the measurement increases with both of them. Furthermore, [46] shows that these errors do not follow a zero-mean Gaussian distribution, but are biased. This bias increases with the range and the incidence angle. Building upon those findings, we can posit that observing from different viewpoints, and observing from a closer range may improve the map quality, as varying the viewpoints also varies the incidence angles from which a surface is observed.

For these reasons, we can hypothesize that the quality of an area of the map is linked to the diversity of the viewpoints and the distance from which it is observed. We will demonstrate this also in this chapter.

5.1.2 Hypothesis Testing

Hypothesis testing is a statistical method aimed at either accepting or rejecting a hypothesis made about the distribution of a population, or at comparing two populations. Often, when examining data, one can formulate a hypothesis based on that data. When comparing two populations, we can use hypothesis testing to determine whether the observed difference between the two populations is statistically significant or merely the result of chance. In this context, the process of conducting a statistical test involves selecting a random sample from each of the two populations we want to compare and then performing measurements and analysis on these two samples. These tests are typically based on what is known as the null hypothesis, denoted as H_0. The null hypothesis posits that there is no relationship between the two sets of sampled data, and any observed difference is purely due to chance. In addition to the null hypothesis, an alternative hypothesis, denoted as H_1, is often defined. The alternative hypothesis suggests that a relationship does exist between the two sets.

To conduct a statistical test, we assess whether the null hypothesis can be accepted or rejected in favor of the alternative hypothesis using a random sample of the dataset. A statistical test involves the calculation of two key quantities: the test statistic, which summarizes the characteristics of each sample, and the p-value, which represents a probability. The p-value quantifies the statistical significance of the result, with lower p-values indicating stronger evidence against the null hypothesis in favor of the alternative.

When the objective is to compare the distributions of two populations, the most commonly used statistical test is the Student's t-test. However, this test assumes that the two populations follows a normal distribution. In our work, such an assumption cannot be made. Therefore, following the approach of [27], we utilize Mann-Whitney U tests [50]. This test is non-parametric, meaning it makes no assumptions about the distribution of the two populations. An implementation of this test is available in the Python library scipy-stats [75].

The Mann-Whitney U test compares the cumulative distribution functions (CDF) of the distribution behind the two populations. Let $F(u)$ and $G(u)$ represent the CDFs of populations x and y, respectively. In this test, the null hypothesis (H_0) is simply that the underlying distribution behind the two populations are equal. The alternative hypothesis (H_1) is selected as either "less" or "greater", depending on the specific hypothesis being tested, as we will show in Section 5.2.2. A small p-value from this test implies that H_0 is rejected in favor of H_1, with a significant result. In other words, it means that the hypothesis that the distributions are equal is rejected in favor of the hypothesis that the underlying distribution behind one population is either less or greater than the other.

5.1.3 Random Forest Regressor

In recent decades, learning from data have seen significant development. Available methods span from the traditional linear regression to more recent techniques like neural networks or random forests. In machine learning terms, the task of learning from data to predict an output based on a set of inputs is referred to as supervised learning. Given a set of input variables X and a set of output variables Y, the objective of supervised learning models is to discover the function $\phi : X \rightarrow Y$, that produces the best predictions, denoted as $\hat{Y} = \phi(X)$. When Y consists of categorical values, the learning task is a classification problem, whereas when Y consists of numerical values, it is a regression problem. The process of learning the model is called "training" whereas the application of the model to make predictions is called "inference".

In this study, our focus is on Random Forests. Random Forests, introduced by [14], are known for their ability to work well with relatively small training datasets, ease of training, and robustness in handling outliers or noisy variables [48]. In essence, a tree is a data structure that organizes data into leaves or nodes. Splitting data from node t consists in dividing the space into subspaces, each corresponding to a child node of t. In a regression tree, the data is split based on thresholds values of input variables. In the specific case of a binary tree, data greater than the threshold is assigned to the right branch, whereas data smaller than the threshold is assigned to the left branch. When the termination criterion is met, the data is no longer split, and the node stores the average output value of the data points it corresponds to. A Random Forest Regressor is an ensemble of regression trees, with each tree incorporating a randomization factor. During training, each tree is constructed with solely a random subset of the dataset, referred to as the bootstrapped dataset. Additionally, each tree does not necessarily consider all the variables, but only a subset of them. These two sources of randomness lead to a diversity of trees in the forest. During inference, predictions are averaged across all the trees.

A fundamental aspect of these trees is how data is split. The decision to split the data is made with the goal of achieving more homogeneous and less variable subsets, a measure commonly referred to as impurity. For regression trees, the most common impurity measure is the Mean Square Error (MSE). The best split is the one that minimizes the impurity.

One noteworthy feature of Random Forests is their ability to assess the importance of input variables, as demonstrated in [48]. In our work, we are interested not only in making accurate predictions, but also in identifying which input variables are the more important in making these predictions.

5.1.4 Importance Measure

The work by [48] presents a comprehensive study of importance measures with Random Forests. Several versions of importance measures have been introduced in the literature and are implemented in the Python Scikit-learn library [56]. In what follows, we provide details about the classical ones, based on impurity measure and permutation importance.

A first approach to define the importance of an input variable is to assess its ability to reduce impurity. The importance of a variable is then the Mean Decrease Impurity (MDI) ([14, 15]). Despite its appealing simplicity, this measure has several drawbacks. Notably, it can inflate the importance of numerical features. In addition, this classical importance measure is computed from statistics derived from the training dataset. It implies that the importances can be high even for features that are not predictive of the target variable, as long as the model has the capacity to use them to overfit. An alternative approach to measure feature importance introduced in [14, 15] is the permutation importance. This consists in measuring the Mean Decrease Accuracy of the forest when the values of a variable are randomly permuted on data not used for training. As demonstrated by [48], the main limitation of estimating feature importance with MDI is that this depends on the training data only, and may be biased if the model is overfitting. Consequently, they recommend permutation importance on a test set. This approach provides an estimate of the importance of input variables at generalization.

5.2 Method

Before going further, it is important to keep in mind that the idea behind what is presented here, is to find a way to predict the map quality, at a local level, directly from the robot's data, without access to any ground-truth. To do so, we have presented in Chapter 3 our framework where we discretize the space into cuboid regions, in order to measure their quality (Chapter 4). In this section, we first present our methodology to compute statistics from the observation viewpoints. Next we propose a methodology to validate that they are predictors of the map quality, on the same cuboid regions. Finally, we show how we evaluate a learned predictor, and how we measure the importance of each variable towards the prediction.

The ROS package computing the viewpoint statistics is available on github [9].

[9] `https://github.com/stephanie-aravecchia/obs-stats-NBV.git`

5.2.1 Statistics from the observation viewpoints

Following our hypothesis expressed in 5.1.1 on factors impacting map quality, we consider four observation viewpoint statistics. For each cuboid region, with every new observation, the viewpoint statistics we consider are:

- n_{obs}, the number of observations;
- r_{min}, the minimum distance of the observations;
- n_Ω, the number of angular sectors covered by the viewpoints;
- σ_θ, the spherical variance of the viewpoints.

The two latter intend to express the diversity of the viewpoints. This section explains how these statistics are computed.

Viewpoint statistics update

To begin with, we describe how the statistics are stored and updated. We first construct and initialize $\mathcal{G}$, a 3D-grid containing observation viewpoint statistics. $\mathcal{G}$ is in the same reference frame R_f than the reconstructed map and the ground-truth, and each element $\mathcal{C}_s$ in this grid spatially corresponds to a cuboid region described previously (same resolution *RES*, same position in R_f). $\mathcal{C}_s$ stores the viewpoint statistics of its corresponding cuboid region. Each time we receive a new point-cloud from the Lidar, we update at most once $\mathcal{C}_s$, the elements of $\mathcal{G}$, to keep track of the viewpoint statistics. We know that each point P of the point cloud is observed from the center of the Lidar, O. The laser ray is $[PO]$. We also know that each element along $[PO]$ has been observed from the same point O. We calculate the intersection of the segment $[PO]$ and the grid $\mathcal{G}$. For each element in this intersection, we update the statistics from the observations, considering the center $C = (x_c, y_c, z_c)$ of $\mathcal{C}_s$ is observed from O. We perform the ray-casting operation to compute the viewpoint statistics, regardless that the cuboid region contains an object or empty space. The update of $\mathcal{G}$ is fast to compute and easily runs in real-time.

In what follows, we consider we are performing the k^{th} update of $\mathcal{C}_s$.

Number of observation n_{obs} and minimum range r_{min}

The objective of both statistics is to store really simple data. Every time a cuboid $\mathcal{C}_s$ is updated, we statistics values as follows:

$$n_{obs} = 1 + n_{obs}^{k-1} \tag{5.1}$$

$$r_{min} = \begin{cases} ||CO||, & \text{if } ||CO|| < r_{min}^{k-1} \\ r_{min}^{k-1}, & \text{otherwise} \end{cases} \tag{5.2}$$

Angular Sectors n_Ω

The objective here is to count the number of angular sectors each cuboid region has been observed from. To do so, we divide the horizontal plane (x, y) going through the center of the considered cuboid region $C = (x_c, y_c, z_c)$ into n angular sectors, represented with Ω, a boolean vector of size m. For each observation from a point $O = (x_o, y_o, z_o)$, we compute the azimuthal angle in the (x, y) plane: $\theta = atan2(y_o - y_c, x_o - x_c)$. From θ, we compute the angular sector index i in Ω: $i = \theta \times m/2\pi$, and set Ω_i to 1. The statistic we propose is simply what follows, with $\vee$ the Boolean OR operator:

$$n_\Omega = \sum_{i=0}^{i=m-1} \Omega_i^k \vee \Omega_i^{k-1} \tag{5.3}$$

Spherical Variance σ_θ

Calculating the spherical variance, defined in [73], consists in encoding each viewpoint through its spherical angle with the coordinate axes, $\mathbf{U} = [U_x, U_y, U_z]$ normalized to a unit vector. Assume now that we are given n observation viewpoints $\mathbf{U(1)}, \cdots, \mathbf{U(n)}$. The spherical variance is defined as follows :

$$\sigma_\theta = 1 - R/k \text{ with } R = \sqrt{X_{sum}^2 + Y_{sum}^2 + Z_{sum}^2} \tag{5.4}$$

where we denote

$$X_{sum}, Y_{sum}, Z_{sum} = \sum_{j=0}^{j=k} U_x(j), \sum_{j=0}^{j=k} U_y(j), \sum_{j=0}^{j=k} U_z(j) \tag{5.5}$$

We compute this spherical variance for each update, where we derive $\mathbf{U}$ from $[CO]$, and then calculate the resultant length R from the history of $\mathbf{U}(k)$, from observation 0 to the current one k, in the considered cuboid region $\mathcal{C}_S$.

5.2.2 Validation of the indicators with statistical tests

The objective of this section is to prove that the observation viewpoint statistics we consider are closely related to the map quality, and to validate that they are indeed relevant indicators

of this map quality. As shown in Chapter 4, we measure the map quality of each cuboid with one of the following metrics. If the cuboid is in occupied space, $\mathcal{U}_{occ}$, we measure the map quality with the Average Hausdorff Distance *AHD*. If the cuboid is in empty space, $\mathcal{U}_{empty}$, the measure is done with the L_1 norm. Since the grids of cuboids are aligned in space, each cuboid region yields a measure of the map quality (*AHD* or L_1), and statistics from its observation viewpoints (n_{obs}, r_{min}, n_Ω, σ_θ). Let M be the value of its measured quality and s the value of the statistic. We split the population of cuboids regions in two populations, using a threshold for the considered indicator, and we perform a statistical test to prove that the map quality M of the population of cuboids for which the indicator s is above (or below) the threshold is significatively better than the other population.

metrics M	indicator s	threshold $s*$	alternative hypothesis H_1
AHD or L_1	$n_{obs}, n_\Omega, \sigma_\theta$	n_{obs}*, n_Ω*, σ_θ*	$M[s \geq s*] < M[s < s*]$
AHD or L_1	r_{min}	r_{min}*	$M[s < s*] < M[s \geq s*]$

Table 5.1: Details of the Mann-Whitney U tests populations and hypothesis

To perform the tests, we set the threshold $s*$ to the median of s. Then, following Section 5.1.2, we compare the two populations with the one-sided non-parametric Mann-Whitney U tests indicated in Table 5.1. We use the scipy-stats library [75]. If the p-value is small, then we reject the null hypothesis H_0, that the two populations of cuboids are equally well mapped, in favor of the alternative H_1, corresponding to a significant better map quality when the value of the indicator of interest is above $s*$ (or below, for r_{min}). In other words, a small p-value proves the considered viewpoint statistic is undeniably an indicator of the map quality.

5.2.3 Coefficient of Determination and Feature Importance

The second method to validate that the viewpoint statistics are indicators of the map quality is through the learning of a regression model and the computation of the coefficient of determination and the permutation feature importance. To do so, we learn a model of the regression $M = f(s)$ with a random forest regressor [14]. We use the Python Scikit-learn library [56], providing numerous tools to analyze the models. We learn a different model for each set ($\mathcal{U}_{occ}$, $\mathcal{U}_{empty}$). We evaluate the models with the R^2 coefficient. R^2 represents the percentage of variance of the output variable explained by the model. The best possible score is $R^2 = 1$. If the model always predicts the mean, $R^2 = 0$. R^2 follows Equation 5.6.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{5.6}$$

Where SS_{res} is the residual sum of squares and SS_{tot} is the total sum of squares. They are computed as follows, with y the observed data, $\hat{y}$ the predicted value, and $\bar{y}$ the mean of the observed data:

$$SS_{res} = \sum_i (y_i - \hat{y_i})^2 \tag{5.7}$$

$$SS_{tot} = \sum_i (y_i - \bar{y})^2 \tag{5.8}$$

Following [48], as in Section 5.1.4, we perform a permutation importance on the test set to compute an estimate of the importance of the input variable. We simply use the implementation *permutation_importance* provided by the Python Scikit-learn library.

5.3 Experiments and Results

As explained in Chapter 4, we measure the map quality with two distinct metrics: *AHD* for cuboids region in occupied space ($\mathcal{U}_{occ}$), L_1 for cuboids region in empty space ($\mathcal{U}_{empty}$). Both of these metrics are distances: the lower the value, the better the quality. The experimental data yielding these results are derived from the WAYPOINTS experiments described in Chapter 3. In these experiments, the robot is driven manually through the same list of waypoints, in RECT and TREES environments. It is worth noting that the only paragraph in this section that does not involve the use of WAYPOINTS is the following one.

5.3.1 Showcasing the angular sectors and the spherical variance

Before going further, we would like to provide a concrete illustration of the connection between viewpoint statistics and map quality. To achieve this, we use a particular set of experiments, where the world contains a single object. In this case, the object is a cross extruded shape. During the experiment, we drive the robot with two distinct driving behaviors, as introduced in Chapter 3:

- **FRONT**: the robot drives in a straight line towards the object,
- **LAT**: the robot drives in a straight line with the object on its side.

These driving behaviors are designed to enhance the effect of the diversity of the viewpoints only on the map. To do so, the duration of the experiment is the same (so is the number of observations), and the minimum range is similar. The results are shown in Figure 5.1. On the top, the figure displays the cuboids: the ground-truth cuboid and the two reconstructed cuboids from those two driving behaviors. We can see that the reconstruction from LAT is

better than FRONT. Due to the occlusion, in FRONT half of the shape is hidden from the robot. On the bottom, the figure displays the value of the viewpoints statistics and the measure of the quality of the cuboids LAT and FRONT. We can see that, as expected, n_{obs} and r_{min} do not vary significantly. As expected also, we see that n_Ω and σ_θ vary significantly, as does the measure of the quality *AHD*. These two statistics are able to gather information invisible to n_{obs} or r_{min}.

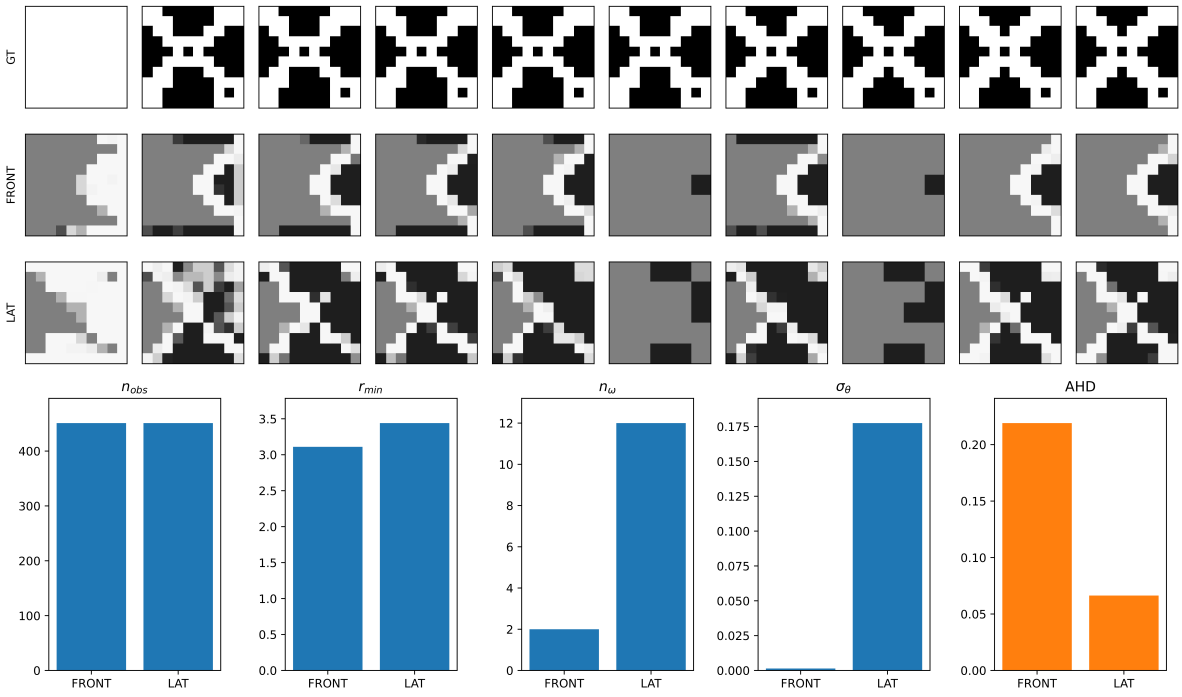

Figure 5.1: Illustration of the impact of the viewpoints on the quality of a cuboid, and on the value of the viewpoint statistics. Top: the cuboids. The first line is the ground-truth, on the second and third lines, the cuboids obtained from two different driving motions (FRONT and LAT). Bottom: the statistics (blue) and quality measure (orange) corresponding to the two cuboids.

5.3.2 Apparent correlation between map quality and viewpoint statistics

In this section, we want to highlight the apparent correlation between the map quality and the viewpoint statistics. To begin with, we simply show the distribution of the data when the quality measure is plotted against a single viewpoint statistic. If the map quality is correlated to the value of the statistic, we should see a trend in the distribution of the data. This is what we see, as shown in Figure 5.2. This figure displays two pieces of information about the data distribution. It displays the median and the Inter Quartile Range (IQR), when the map quality,

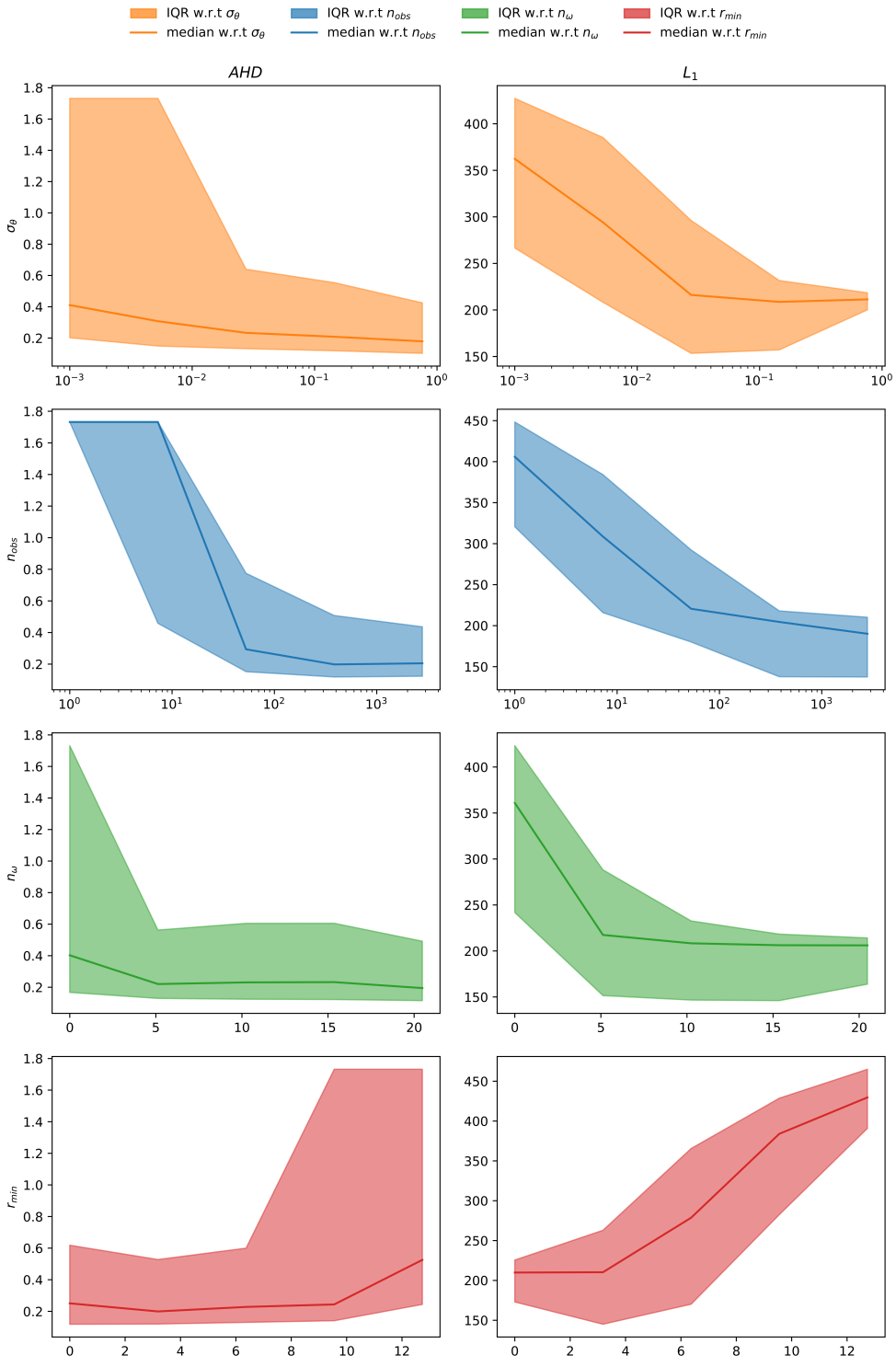

Figure 5.2: Distribution of the map quality against the viewpoint statistics. The first row corresponds to cuboids in $\mathcal{U}_{occ}$, the second row to cuboids in $\mathcal{U}_{empty}$. Each column correspond to a different viewpoint statistic: σ_θ, n_{obs}, n_Ω, r_{min}. The line corresponds to the median value of the metric against the viewpoint statistic, the filled area corresponds to the IQR of the value against the viewpoint statistic.

measured with *AHD*, is plotted against the different viewpoint statistics.

Following what could seem intuitive, we see that the map quality increases when n_{obs}, n_Ω or σ_θ increase (the viewpoints are more numerous / more diverse). Conversely, it increases when r_{min} decreases (seen from closer). This remains true both in occupied and empty space. What is also interesting, particularly in occupied space, is that the spread of the values, shown by the IQR, shrinks when n_{obs}, n_Ω or σ_θ increase and when r_{min} decreases.

Those curves suggest that the map quality may be correlated with the viewpoint statistics. In other words, the viewpoint statistics we consider may be predictors, or at least indicators, of the map quality, and they become better predictors when their value is large (or small, for r_{min}). For completeness, it should be mentioned that the results we display here corresponds to the experiments with a perfect localization. The trend remains the same when the localization is noisy.

5.3.3 Validation with hypothesis testing

To prove that the trend we see is statistically significant, and not simply due to chance, we perform the statistical tests described in 5.1.2, for each viewpoint statistic individually.

Those tests are conducted on the cuboids generated from 18 experiments. Specifically, we utilize the WAYPOINTS experiments. In three of them, the world is TREES, in three others, the world is RECT, and finally, the maps are built in all those experiments using three distinct level of noise in the localization. Ultimately, for each experiment, we perform the 8 statistical tests detailed in Table 5.1. These tests can be summarized as follows:

- if n_{obs}, n_Ω or σ_θ (individually) are higher than their respective medians, the quality is better (*AHD* smaller in $\mathcal{U}_{occ}$, L_1 smaller in $\mathcal{U}_{empty}$) compared to when they are not;
- if r_{min} is less than its median, the quality is better (*AHD* smaller in $\mathcal{U}_{occ}$, L_1 smaller in $\mathcal{U}_{empty}$) compared to when it is not.

Before presenting the results of the statistical tests, we provide details on the size of the cuboid populations on which the tests are conducted. As explained in Chapter 4, we measure map quality only for cuboids that are observed sufficiently. Specifically, we require that at least one voxel within the cuboid has an occupancy likelihood that is different from the unknown by at least 0.1. Consequently, the number of cuboids in occupied or empty space varies across experiments. These variations are influenced by factors such as the trajectory, the type of world, and to a lesser extent, the noise in localization.

Table 5.2 provides the population sizes used in the tests, with a mean size of 6 420 cuboids.

	TREES	RECT	ALL
$C \in \mathcal{U}_{occ}$	1492 ± 347	705 ± 132	1099 ± 479
$C \in \mathcal{U}_{empty}$	6008 ± 1069	4364 ± 801	5321 ± 1157
Number of C	7500 ± 1401	5339 ± 921	6420 ± 1600

Table 5.2: Summary of the size of the cuboid populations used in the statistical tests: mean and standard deviation of the occurrence of cuboids in $\mathcal{U}_{occ}$ or $\mathcal{U}_{empty}$ for the TREES or RECT experiments.

Finally, we present the results of the statistical tests, in Table 5.3, using a common threshold for the p-value: 0.05. In this table, each cell provides the ratio of the statistical tests with a p-value smaller than 0.05, across the 18 experiments. A test with $p < 0.05$ rejects H_0 in favor of H_1 with a strong confidence. In other words, a test with $p < 0.05$ statistically validates our hypothesis, that the viewpoint statistic considered in the test is an indicator of the map quality.

	$\mathcal{U}_{occ}$	$\mathcal{U}_{empty}$
n_{obs}	0.83	1.0
r_{min}	0.62	1.0
n_{Ω}	0.67	1.0
σ_{θ}	0.81	1.0

Table 5.3: Ratio of the experiments for which the p-value of the test is smaller than 0.05

In this table, we can observe that all the tests unanimously agree for the cuboids in $\mathcal{U}_{empty}$: each viewpoint statistic individually is an indicator of the map quality in empty space. Focusing on $\mathcal{U}_{occ}$, we can see that over 81% of the tests confirm that n_{obs} and σ_{θ} individually are indicators of the map quality. The results are slightly lower for r_{min} and n_{Ω}, but still in more than 62% of the tests, our hypothesis is validated. That holds true, even in challenging environments and with noise in the localization.

It is important to note that there does not appear to be any trend in the tests where $p > 0.05$, they are distributed across the two types of assets and the different noise levels.

In these results, the threshold $s*$ is independently computed for each experiment as the median of the viewpoint statistic. The fact that some tests fail to validate our hypothesis does not necessarily imply that the viewpoint statistic is incapable of indicating map quality, but rather that it may not do so effectively with the currently considered threshold. We firmly believe that combining these viewpoint statistics can lead to actual predictions of map quality, and we will explore this further in the following sections.

5.3.4 Learning to predict map quality from viewpoint statistics

This section serves two objectives. The first one is to evaluate the predictive capability of combined viewpoint statistics, while the second is to measure the importance of individual statistics in this prediction. We proceed following the approach detailed in Section 5.2.3.

Dataset

In this evaluation, we focus on a specific subset of cuboids in the map: the cuboids that are at the same height as the robot's Lidar. Since the experiment was conducted with an Ouster-16 Lidar, only the cuboids at the robot's height, and seen from a close enough range, are traversed by several Lidar planes. This is because the Ouster-16 has a vertical field of view of 45 degrees (-22.5, +22.5), covered by only 16 planes. This introduces a bias in the data, which we want to exclude from our evaluation. We illustrate the impact of the cuboid's height in the map, denoted as z, on the distributions of the data (the cuboid's quality) in Figure 5.3. It is evident in this figure that the distributions are notably different for the three "slices" in $\mathcal{U}_{occ}$ (blue) and $\mathcal{U}_{empty}$ (orange). In future work, it would be straightforward to include this z parameter for predicting map quality when working with a ground-robot. By design, this parameter is known for each cuboid.

From now on, we exclusively consider the cuboids where $z = 0$. As a result, we now have two datasets: D_{occ} in $\mathcal{U}_{occ}$ with 6 819 examples and D_{empty} in $\mathcal{U}_{empty}$ with 39 620 examples. We split each dataset into training and test sets. Specifically, we randomly allocate 90% of each dataset into training sets, while the remaining 10% of each dataset constitute the test sets. We normalize both the input and output of the training set using min-max normalization, and we apply the same normalization parameters to the test set. It is important to note that we do not employ any data augmentation technique at this stage.

Training

Following this, we train a Random Forest Regressor on the training set. To optimize the model performance, we perform a grid search on the parameters to find the set of parameters producing the models that generalizes best. From this grid search, the hyperparameters are:

- $max\_depth = None$,
- $n\_estimators = 200$,
- $max\_features = 3$,
- $min\_samples\_leaf = 1$,
- $min\_samples\_split = 2$.

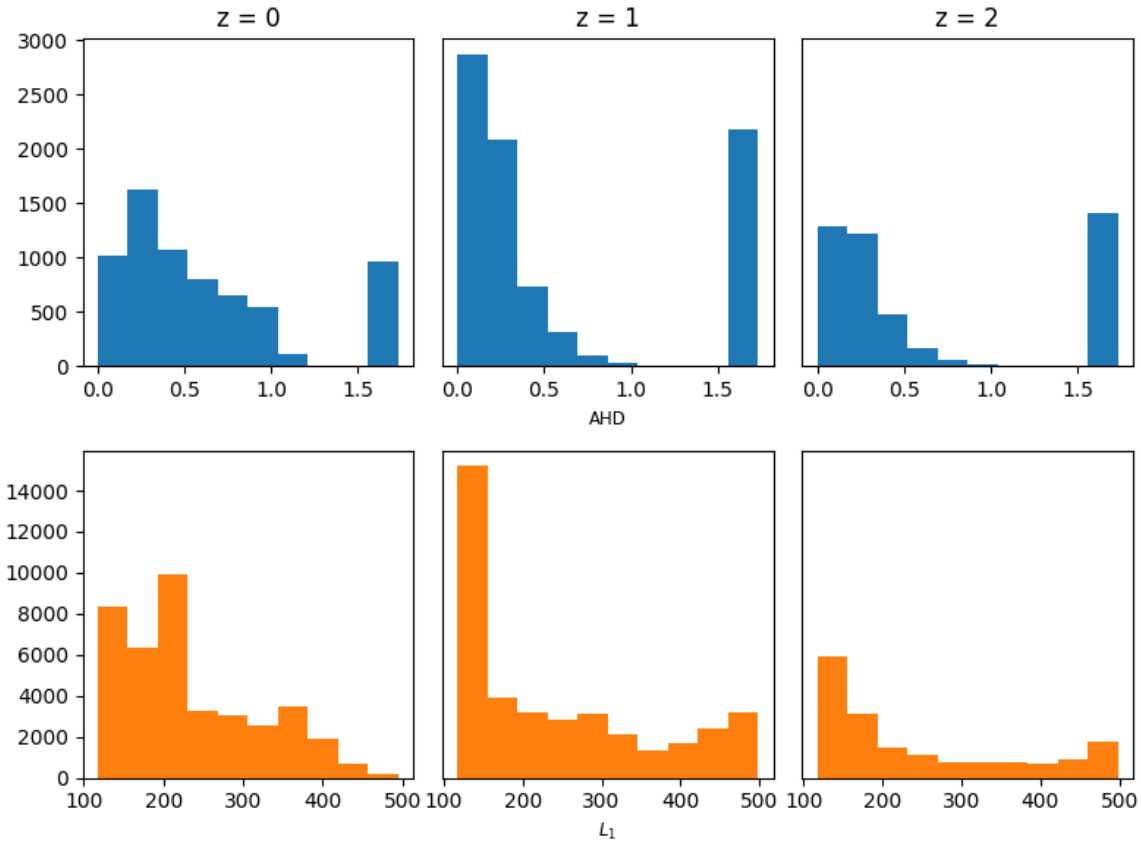

Figure 5.3: Distribution of the map quality depending on the height z of the cuboid in the map. Top: cuboids in $\mathcal{U}_{occ}$. Bottom: cuboids in $\mathcal{U}_{empty}$. One column per value of z.

Model evaluation

Following Section 5.2.3, we evaluate the models with the R^2 coefficients. In the $\mathcal{U}_{occ}$ dataset, the model achieves a score of 0.92 on the training set and 0.42 on the test set. This demonstrates the feasibility of learning to predict quality from viewpoint statistics.

With this experiment, we aim to address two key questions:

- does the type of world affect the learning process?
- is the importance of input variables influenced by the type of world?

To address these questions, we proceed by splitting each dataset into two subsets based on the type of world: RECT and TREES. We maintain the same hyperparameters to ensure consistency in our analysis.

Figure 5.4 shows that the type of worlds indeed affects the learning. This figure displays the R^2 scores of models on the divided datasets. While the model performs well on training data in both environments, it generalizes better in RECT environments compared to TREES environments. This trend is evident in both $\mathcal{U}_{occ}$ (in blue) and, somewhat surprisingly, in $\mathcal{U}_{empty}$ (in orange).

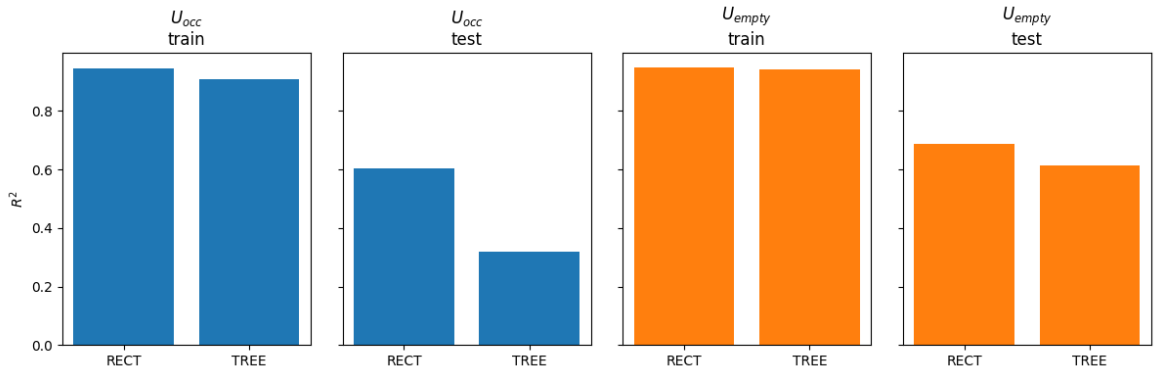

Figure 5.4: Models evaluation with R^2 coefficient on the training and test sets, when learning to predict quality from viewpoint statistics. Blue: in $\mathcal{U}_{occ}$, orange: in $\mathcal{U}_{empty}$. In each graph, each bar corresponds to a different type of environment (RECT or TREES).

Feature importance

Then, to answer the second question, we perform a permutation importance analysis on the trained models, evaluating them on the test sets, as explained in Section 5.2.3. The results, presented in Figure 5.5, clearly reveal a difference in the importance of the statistics between the types of environment, particularly in $\mathcal{U}_{occ}$ (blue). In $\mathcal{U}_{occ}$, n_{obs} is by far the most important feature in RECT environments, whereas in TREES environments, all viewpoint statistics are

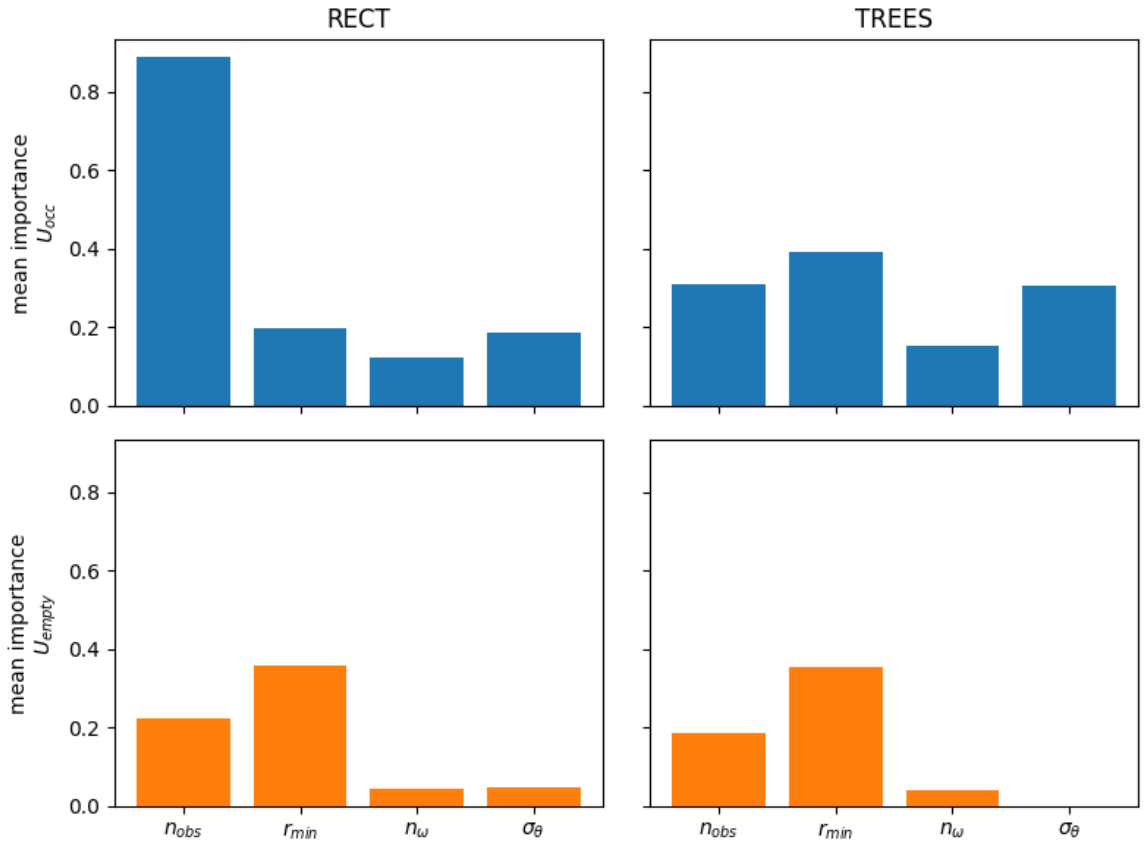

Figure 5.5: Importance of the input variables to the generalization of the model. Blue: in $\mathcal{U}_{occ}$, orange: in $\mathcal{U}_{empty}$. In each graph, each bar corresponds to a different input (the viewpoint statistics).

important. In $\mathcal{U}_{empty}$ (orange), r_{min} remains the most important variable in both environments. This may be attributed to the Lidar used in this experiment, as only cuboids within close range are traversed by multiple planes. The quality of an empty cuboid is directly linked to the rays that have traversed it, the more the rays, the better the quality.

Finally, it is worth noticing that some features are likely to be correlated. As the robot moves in its environment around objects, it is reasonable to anticipate a correlation between n_{Ω} and σ_{θ} with n_{obs}. Cuboids observed from multiple viewpoints are likely to be seen frequently. Likewise, we can expect that a cuboid is more likely to be observed from a closer distance (smaller r_{min}) when it has been seen more frequently. We calculate the Spearman's correlation coefficient on the viewpoint statistics from the dataset. We show the resulting correlation matrix in Figure 5.6. This confirms that n_{Ω} and σ_{θ} are indeed positively correlated with n_{obs}, whereas r_{min} is negatively correlated with the other features.

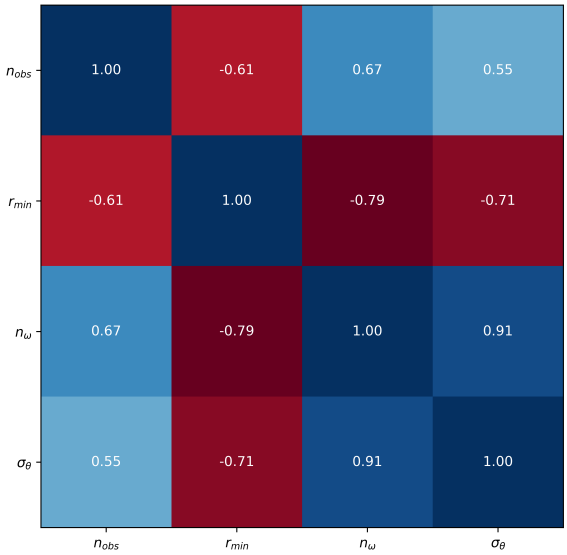

Figure 5.6: Correlation Matrix of the viewpoint statistics in the dataset

Confusion Matrix

Finally, to provide a different perspective in the model evaluation, we compute the confusion matrix on its predictions. To achieve this, we treat our problem as a classification task, discretizing both the true and predicted values into 10 distinct classes. Each class represents a specific range of values, with the same ranges used for both the truth and predicted values.

This confusion matrix displays the occurrence of positive values, whether true or predicted, within the class corresponding to the selected range. Rows correspond to the true class, while columns correspond to the predicted class. Figure. 5.7 displays the confusion matrix computed on the test set for both RECT and TREES environments.

This matrix allows insights into the model prediction's errors. One notable observation is that the majority of data points are concentrated in the top-left corner, which aligns with the data distribution shown in Figure 5.3. Additionally, what is particularly noteworthy, is that the positive values tend to cluster near the diagonal. The closer a point is to the diagonal, the smaller the prediction error. This suggests that while the model may not reliably predict the exact quality value of the map from the statistics, it does provide a reliable indication of the map's quality, both in RECT and TREES environments.

5.4 Summary

In this chapter, we firstly introduced four observation viewpoint statistics, specifically, the number of observations (n_{obs}), the minimum range (r_{min}), the number of angular sectors (n_Ω),

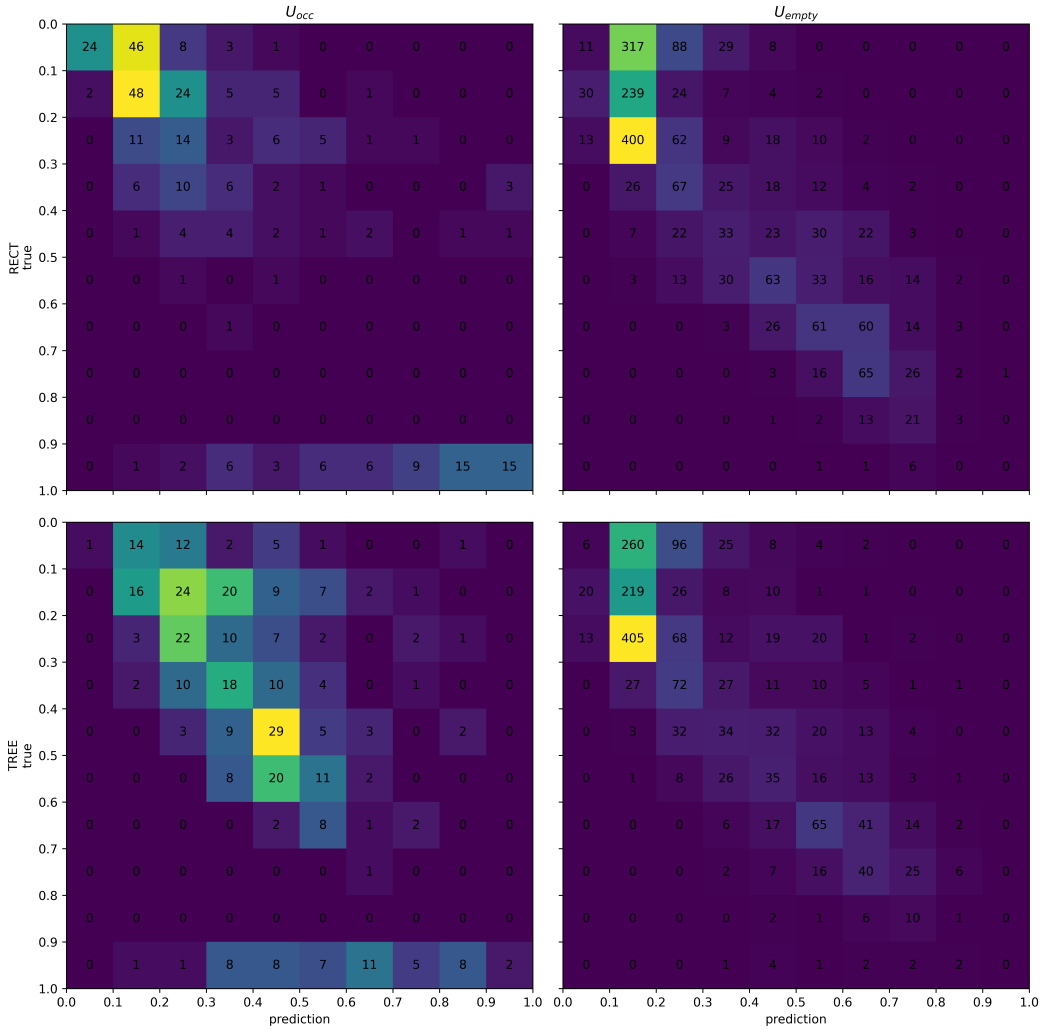

Figure 5.7: Confusion Matrix that display the occurrence of true and predicted values on the test set. Right: in $\mathcal{U}_{occ}$, left: in $\mathcal{U}_{empty}$. The color scale is normalized for each matrix, from purple 0, to yellow max.

and the spherical variance (σ_θ). We explained how to compute and continuously update them in a 3D grid, allowing us to select cuboids, as discussed in Chapter 3, to measure their quality, as in Chapter 4, and to search for a link between quality and statistics.

Secondly, we presented a methodology to validate the viewpoint statistics as indicators of local map quality through rigorous statistical tests. This validation process was intended to confirm that an observed trend was not mere chance but consistent and meaningful reflection of the data.

Thirdly, we proposed a methodology for evaluating the relative importance of each statistic in predicting local map quality. It consisted in training a Random Forest Regressor to learn the relationship between quality and statistics, and in performing a permutation importance

on the input variables, namely, the viewpoint statistics.

Regarding our findings, we showcased the correlation between map quality and viewpoint statistics through quality vs. statistics plots. These plots visually represented the relationship between these variables. Furthermore, we validated the statistical significance of the observed correlations using hypothesis testing.

Additionally, our experiments revealed that learning in structured environments is comparatively less challenging than in unstructured ones. Notably, in unstructured environments, a combination of statistics, including n_{obs}, r_{min}, and a statistic reflecting viewpoint diversity (e.g., n_Ω or σ_θ), is essential for accurate prediction, whereas n_{obs} only may be sufficient in structured environments.

6

Autonomous exploration with View-Point Statistics based policy

The work presented in the previous chapters aimed to lay the groundwork for the research presented here. Our objective is to tackle an exploration task in natural environments, with the objective to enhance map quality. In this chapter, we integrate the view-point statistics introduced in Chapter 5 in the selection of the Next-Best-View. While this chapter outlines the final objective of this work, and provides initial insights to solve the task, it serves a foundational step for future research.

Unlike studies focusing on large environments with significant, relatively few prominent features, such as plants for instance, the work presented here is centered on large, sparse environments characterized by numerous small objects scattered throughout. In these environments, the areas of interest are both widespread throughout the environment and comparatively few in number. In Chapter 5, we demonstrated the four viewpoint statistics we consider (number of observations, minimum range, number of angular sectors, spherical variance) were indicators of map quality in such environments.

In this work, we construct exploration policies based on these viewpoint statistics individually. Specifically, we calculate the information gain of candidate NBVs using these statistics. Our results show that these policies outperform baselines in improving map quality during the exploration process, with more pronounced differences in the type of environment we consider.

6.1 Related Work

6.1.1 Exploration Policies

In Chapter 2, we presented the concept of exploration. Exploration policy is essentially the decision-making process that guides a robot in selecting its next destination in an unknown environment. Within this chapter, we also introduced the notion of frontier points: points within the known volume with at least an unknown neighbor. From this foundational concept, two traditional exploration policies are derived: the closest-frontier policy, where the robot's goal is the nearest reachable frontier point, and the random-frontier policy, where the robot selects is goal randomly amongst the reachable frontier points.

Expanding on the idea of frontier points, numerous research papers have explored strategies to rapidly cover the volume of the environment. The primary aim in such approaches is to reduce the unknown areas on the map swiftly, without necessarily considering the confidence level in the newly discovered regions. For instance, in [9] they propose setting goals by filtering and clustering frontier points at different levels in Octomap's octree.

Often, when it comes to cover rapidly the volume, research focuses on solving the ex-

ploration task with a team of robots. While the field has evolved, [37] offers a comprehensive review of the challenges associated with exploration, whether conducted with or without robot collaboration. Recently, in [82], they propose to solve quickly the exploration with a homogeneous team of robots. They use a fleet of UAVs to perform a rapid exploration task in a decentralized way. Conversely, multi-robot heterogeneous exploration strategies have emerged, as proposed to solve the DARPA Subterranean (SubT) Challenge [10]. To solve the challenge, [44] explores with a team of legged robots and UAVs, whereas [80] explores with a team of tracked robots and UAVs.

6.1.2 Next-Best-View Policies

As also introduced in Chapter 2, Next-Best-View (NBV) is an active research topic. NBV, or active sensing, consists in moving the robots in order to maximize the efficiency of the perception.

NBV is not exclusively linked to exploration. In many cases in the literature, NBV is associated with improving the accuracy of reconstruction, often without considering exploration. For example, in [51] and in [43], NBV selection aims to create high quality surfaces for single small-scale objects. However, these methods involve computationally expensive calculations, such as estimating real-time surfaces and their normals, preventing them from scaling to large environments. The scaling is tackled by [2] who chooses NBV to enhance 3D-reconstruction quality of large and complex structures. This is done after an initial scan is performed to obtain a rough model of the structure to be reconstructed.

In the context of exploration, NBV can also serve the purpose of rapidly covering the volume. For instance, in [64], NBV selection is based on the potential volume that can be discovered from the candidate viewpoints, calculated using ray-casting operations. Furthermore, NBV selection is often associated with inspection tasks. For instance, it can aim to maximize information gathering on a surface manifold, as demonstrated by [58] on a 2D-plane, or [84] on a 3D-surface. In both cases, they consider uneven distribution of information, and the goal is to maximize information gathering using Gaussian Processes to encode spatial correlations.

Differently, [36] focuses on exploring a scene containing an object of interest while improving reconstruction quality. They use a visual odometry algorithm to create the 3D-point cloud, and their various NBV selection aim to choose the most informative view to the reconstruction. To make this selection, they use the volumetric information of the voxels. It consists in encoding the level of information of each voxel through the distance between its

[10] `https://www.darpa.mil/program/darpa-subterranean-challenge`

probability measure and the probability measure corresponding to the unknown (i.e. its entropy).

Closer to our work, [68] proposes to select the NBV to improve the quality of the reconstructed surface. Their approach takes into account both the volumetric map and the quality of the reconstructed surfaces, using TSDFs (Truncated Signed Distance Fields). TSDFs enables real-time estimation of a surface point cloud. By averaging weights from neighbor points, they estimate a confidence level associated with each point. Building upon [68], [34] extends the methodology for a fleet of UAVs. However, these methods are primarily designed for structured environments with a single dominant feature to map, such as a plant, as evaluated in their studies. In unstructured environments with sparse Lidar data and localization noise, estimating surfaces using TSDFs may not yield satisfactory results.

One significant limitation of these methods is that they assume that the object of interest is relatively large compared to the scale of the scene. This assumption is the main drawback, as it makes these methods less suitable for large-scale, sparse, and unstructured environments, each of which poses unique challenges. In such environments, not only are there few objects to reconstruct in vast scenes, but there is also considerable Lidar noise, the environment can behave semi-transparently, the object sampling in the 3D grid is non-uniform, and unpredictable occlusions are frequent. In contrast, our method operates without imposing such restrictive assumptions. Our NBV policies rely solely on observation viewpoint statistics, eliminating the need for assumptions about the scene or object to be reconstructed.

Lastly, it is worth noting a common drawback of NBV: the selection of the best goal often involves computationally expensive processes. NBV selection frequently requires intensive computations, such as ray-casting along the path to the goal for numerous potential goals and multiple path computations to those goals. Many papers have aimed to find efficient algorithms to address this challenge, including approaches based on Rapidly-Exploring Random Trees (RRT [47]) or their variant RRT* [39], like [13] or learned methods using Convolutional Neural Networks (CNN), like [77]. Among the latest, [62] introduces a novel approach based on deep reinforcement learning. In this study, we set aside this computational challenge, leaving that to later work, as we will discuss in more detail in Section 6.4.

6.1.3 Policies

In the preceding sections, we referred to policies without explanations. This paragraph is dedicated to providing the necessary context. A policy corresponds to a strategy, to the robot behavior. It outlines what the robot should do, in a given state, specifying the next action or sequence of actions it should take. In an exploration context, this concept is introduced by

[70]. They propose a method for computing the information gain associated with an action, which guides the robot from its current location to a goal location. In this case, the expected information gain represents the expected change of entropy in the map when executing that specific action. Then, the choice of the action is based on the calculation of its expected utility. The expected utility of an action is a balance between the expected information gain and the cost of that action. Our methodology builds upon this method.

What is worth noting is that this approach is very similar to a Partially Observable Markov Decision Process [38], also known as POMDP, with a one step look-ahead only. A Markov Decision Process [59] (MDP) models an agent interacting with a world in a synchronous manner. An MDP is a mathematical framework defined by its components: (S, A, P, R, γ). Where S is the set of states, A the set of actions, T the transition function $T(s'|s, a)$, R the reward function $R(s, a)$, and γ the discount factor. In that context, a policy π is a function that maps from states to actions. The objective of an MDP is to find the optimal policy $\pi*$, that will maximize a quantity called the utility U . $U^{\pi}(s)$ corresponds to the current reward and all the expected rewards in the future if we choose the policy π. When there is uncertainty in the state, that is, we do not have access to the true state, but only a belief of the state using observations, it becomes a POMDP. MDP and POMDP are the core of several Reinforcement Learning algorithms, such as MuZero [66] or Dreamer [33].

6.2 Method

This paragraph presents our method where the NBV selection is based on viewpoint statistics. This approach builds on the work described in Chapter 5, where we construct a grid of viewpoint statistics denoted as $\mathcal{G}$.

As discussed in the previous section, the policy determines the strategy for selecting the NBV. In this study, we specifically focus on identifying the next best action rather than best sequence of actions. In this context, the terms "best policy" and "best next action" are interchangeable, representing a balance between the expected information gain and the cost associated with the next action.

In this section, we first explain what constitutes an action, and how the action's cost is calculated. Following that, we detail the computation of the expected information gain, which relies on a single statistic: either n_{obs}, r_{min}, n_{Ω} or σ_{θ}. Next, we explain the policy itself. And finally, we detail our evaluation methodology, in which we compare our four policies with traditional approaches to assess their effectiveness in improving map quality.

The ROS packages to compute the costmap and our policies are available on github [11].

[11] `https://github.com/stephanie-aravecchia/obs-stats-NBV.git`

6.2.1 Costmap and Candidate NBV

We start by aligning our problem with the 3D-grid of viewpoint statistics G. Since our robot operates on the ground, we focus solely on candidates in the ground plane. Consequently, the 2D-grid of candidates is the ground projection of G. To refine our selection, we define the candidate NBVs, or targets, T. These targets encompass all the reachable candidates, within the known free space, and are restricted to a specified range around the robot's current position, R.

For each target candidate T, let us consider a_T the action of moving the robot from R to T. The cost $cost(a_T)$ is a function of the distance the robot has to travel from R to T, while avoiding obstacles. We calculate these costs with a Breadth-first search algorithm, and store them into a costmap. We consider the cost of moving to a direct neighbor constant, and denote it c. Starting from the cell that corresponds to the robot's current position (the root node), we propagate the displacement cost c to its neighboring free cells. There are at most eight neighbors (the children nodes). In the costmap, cells corresponding to obstacles are marked with the maximum cost value, while a distinct value marks unknown space.

6.2.2 Expected Information Gain

Then, for each candidate NBV, T, we calculate the expected information gain $E[I(a_T)]$. As a reminder from Chapter 5, $\mathcal{G}$ is the 3D-grid containing observation viewpoint statistics. The elements of $\mathcal{G}$ are denoted $\mathcal{C}_s$ and store the viewpoint statistics of their corresponding cuboid region.

Firstly, we consider the current state, $\mathcal{G}_0$, and the predicted state $\hat{\mathcal{G}}$. We compute $\hat{\mathcal{G}}$ by updating every visible cuboid $\mathcal{C}_s$ of $\mathcal{G}_0$ with a new expected observation after simulating taking a_T (i.e. a new observation from T). For n_Ω, we update all the angular sectors in Ω_i covered by the displacement of the robot from R to T. For the three other statistics, we simply consider the new observation from T.

Secondly, given the two states $\mathcal{G}_0$ and $\hat{\mathcal{G}}$, $E[I(a_T)]$ is the accumulation of the considered statistic's gains over all visible cuboids. Given a cuboid C_s, let us consider two quantities s_0^i and $\hat{s}^i$, to obtain a third one g^i:

- s_0^i corresponds to an indicator in $[n_{obs}, r_{min}, n_\Omega, \sigma_\theta]$ in state $\mathcal{G}_0$,
- $\hat{s}^i$ corresponds to the updated indicator in state $\hat{\mathcal{G}}$,
- g^i is the indicator gain.

We set the indicator gain g^i as the surprisal. The surprisal for discrete-time systems is defined in [65] as the logarithmic deviation from the current state. Following that, we compute

g^i as follows:

$$g^i = \begin{cases} log(\hat{s}^i) - log(s_0^i), & \text{if } \hat{s}^i > s_0^i \text{ and if } s \in [n_{obs}, n_\Omega, \sigma_\theta] \\ ||log(\hat{s}^i) - log(s_0^i)||, & \text{if } \hat{s}^i < s_0^i \text{ and if } s \in [r_{min}] \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

Finally, for one statistic, the expected information gain is computed as follows, where g^i depends on the considered statistic, and n is the number of visible cuboids from T:

$$E[I(a_T)] = \frac{1}{n} \sum_{i=1}^{i=n} g^i \tag{6.2}$$

6.2.3 NBV selection policy

Following [70] presented in Section 6.1.2, we select the NBV based on two quantities: $E[I(a_T)]$, the expected information gain of taking a_T, and its associated cost $cost(a_T)$.

We strictly apply their method. To begin with, we calculate $E[U(a_T)]$, the expected utility of a_T, with α a weighting factor:

$$E[U(a_T)] = E[I(a_T)] - \alpha \times cost(a_T) \tag{6.3}$$

Finally, we select the NBV, which is the candidate T, for which a_T* is the action with the highest expected utility:

$$a_T* = argmax(E[U(a_T)]) \tag{6.4}$$

6.2.4 Evaluation Methodology

The underlying idea here is to evaluate how the NBV policy influences map quality. We expect that our policies would lead to better maps compared to baseline policies.

Map quality

To enable this comparison, we firstly need to evaluate the map quality globally. To achieve this, for each experimental exploration, we consider $\mathcal{Q}$, which represents the map quality of the complete volume. $\mathcal{Q}$ is the weighted mean of the normalized metrics calculated on the cuboids regions of $\mathcal{U}_{occ}$ and $\mathcal{U}_{empty}$.

If $\mathcal{U}_{occ}$ and $\mathcal{U}_{empty}$ contain respectively n and m elements, $\mathcal{Q}$ is calculated as follows:

$$Q = \frac{1}{n+m}\left(\sum_{k=0}^{k=n}\frac{cov_k - \min(cov)}{\max(cov) - \min(cov)} + \sum_{k=0}^{k=m}\left(1 - \frac{L_{1_k} - \min(L_1)}{\max(L_1) - \min(L_1)}\right)\right) \quad (6.5)$$

It should be noted that in this particular study, we measure the map quality in $\mathcal{U}_{occ}$ with the surface coverage *COV*. In future work, we will measure this with *AHD*, following our conclusions from Chapter 4.

Finally, considering the context of exploration, our aim is to compare the various policies based on how map quality evolves throughout the exploration process. Specifically, we are interested in how $\mathcal{Q}$ changes as the amount of unknown space in the map decreases. To perform this comparison, we will create plots that showcase $\mathcal{Q}$ against the ratio of discovered space.

Baselines

We compare our four NBV selection policies, where the information gain is based on n_{obs}, r_{min}, n_Ω or σ_θ (6.2), with three other goal selections:

- *random-frontier*: next goal is randomly sampled on reachable frontier point,
- *closest-frontier*: next goal is nearest reachable frontier point,
- *random-free*: next goal is randomly sampled in free space.

While the first two baselines, *random-frontier* and *closest-frontier*, are traditional policies that were presented in 6.1.2, *random-free* is specifically designed for this evaluation. Our policies may sample goals in the known space, and it may not be fair to solely compare them against frontier algorithms that exclusively sample goals in the frontier area adjacent to the unknown space. Introducing a policy that randomly samples goals inside the known space helps remove this potential bias, in our opinion.

6.3 Experiments and Results

6.3.1 Experiments

The experiments are the TELEPORT experiments described in Chapter 3.

We use 12 simulated environments (3 different spatial distributions, with 4 different assets). The spatial distributions are sparse. For each environment, for each of the 3 noise levels, we run 3 simulations with each policy:

- *random-frontier*,
- *closest-frontier*,
- *random-free*,
- ours[n_{obs}],
- ours[r_{min}],
- ours[n_{Ω}],
- ours[σ_{θ}].

Each experiment consists in teleporting the robot to the goal, accumulating the point-cloud, and teleporting it to the next goal. We repeat until the ratio of discovered space in the ground-plane reaches 70% or after 100 teleportations, whichever comes first.

In total, we run 756 experiments in simulation.

6.3.2 Results

Illustration of the components

We would like to begin by presenting an illustration of an experiment in Figure 6.1. In this figure, there are two notable features: the costmap and the candidates.

The costmap, covering the complete figure, encodes various types of information. The grey color represents the unknown space, while other colors represent different costs. Pink corresponds to areas that are lethal for the robot (obstacles), and the other cells range from blue (indicating lower cost) to red (indicating higher cost). We see in this figure that the cost increases when the cell is farther away from the robot, marked by the red square, while avoiding pink cells.

The candidates are represented by the green squares, which are spread within a certain range from the robot's position. The information gain associated with each candidate is encoded through the alpha channel of the green squares, with more transparency indicating lower information gain and less transparency indicating higher information gain.

Finally, the best candidate, determined by the policy, is marked as the yellow sphere.

Policy evaluation

Fig. 6.2 displays how the map quality, denoted as Q, evolves throughout the exploration phase as more space is discovered. These results are presented for experiments conducted in four different simulated environments, where the spatial distribution of assets is consistent, but the specific assets vary. These results only include experiments with perfect localization (84

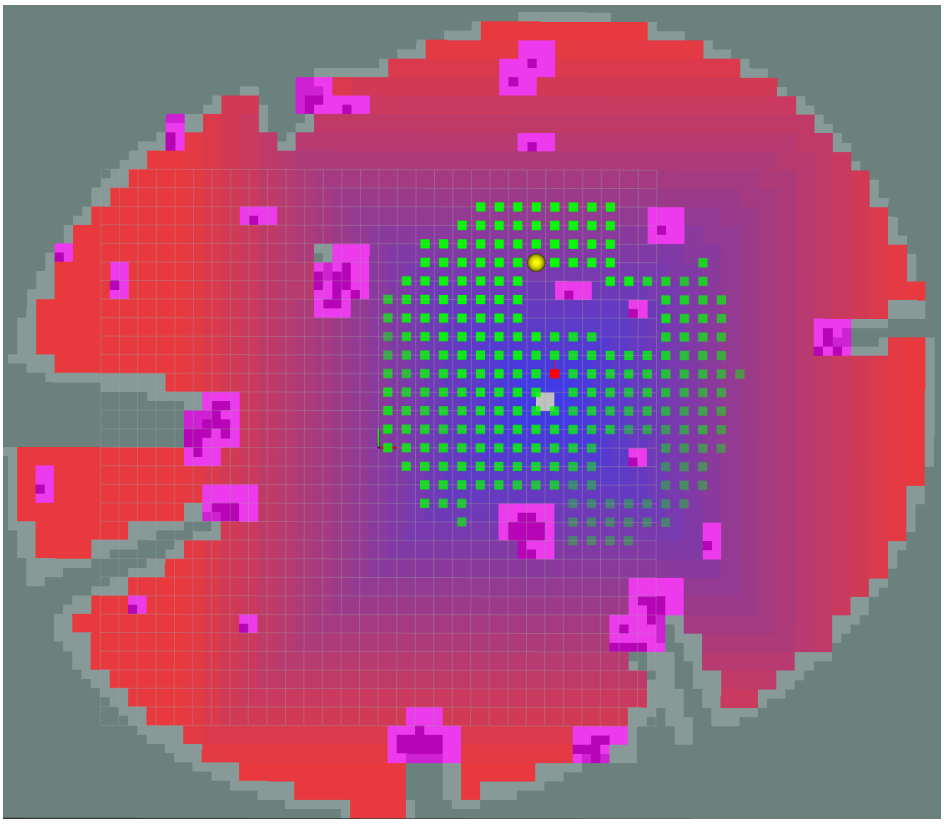

Figure 6.1: Visualization of an experiment, showing the costmap (the grey, pink, red and blue image), the candidates (the green squares), their information gain (the transparency of the green squares), and the best candidate (the yellow sphere).

experiments). Notably, the trends observed in the curves remain consistent regardless of localization noise or asset distribution.

In general, our methods consistently exhibit higher curves compared to the others. This difference becomes more pronounced as the difficulty in the assets increase. These curves demonstrate that developing NBV policies based on viewpoint statistics significantly enhances map quality during the exploration process. This improvement is especially notable in more challenging environments.

However, we believe this enhancement could be further emphasized by formulating the information gain using all four statistics rather than just one. This approach presents certain challenges, particularly in terms of normalization and weighting of the different statistics. To address this, one potential solution would be to formulate the information gain in relation to map quality. Indeed, we introduced in Chapter 5 that map quality can be predicted directly from the statistics.

Additionally, it is worth noting that in these experiments, the robot is teleported. While this solution may find applications, such as selecting NBV to scan an area with a total station,

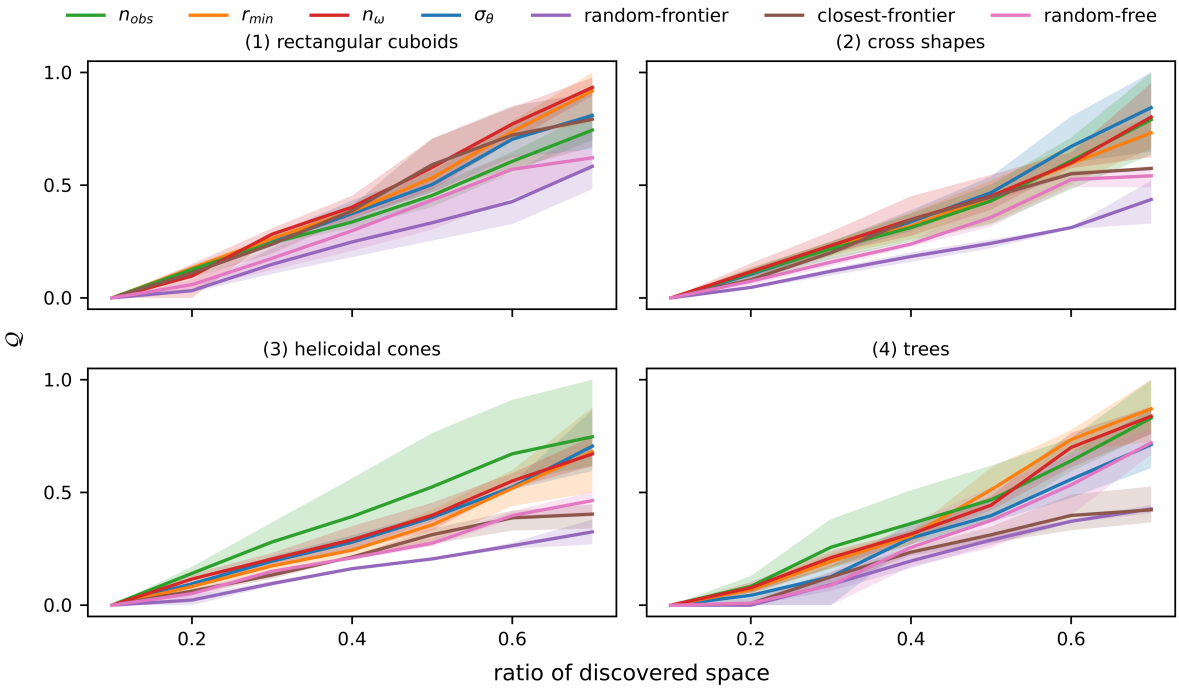

Figure 6.2: Quality of reconstruction expressed as a function of the proportion of discovered space in the ground plane. The plots are arranged with an increasing difficulty in the scene. In each figure, the line is the mean of the experiments, the area is between the min and the max. Our proposed methods ($n_{obs}, r_{min}, n_\Omega, \sigma_\theta$) are compared to the baselines (*random-frontier*, *closest-frontier*, *random-free*).

it does not fully address the problem we aim to solve. In real exploration scenarios, the robot observes during the execution of the path. In this case, computing the expected information gain for all candidates, considering viewpoints during the (potential) path to the candidates, is not feasible in terms of computation time. Therefore, we plan to explore solving the task with Reinforcement Learning, as we introduce in the next section.

6.4 Preliminary Study on Reinforcement Learning

In this section, we present the work we have started to improve the exploration policy with Reinforcement Learning, and particularly using MuZero [66]. In this section, we first introduce quickly reinforcement learning, we then discuss the preliminary work that has already been conducted, and we finally outline our methodology concept and our experimental plan.

6.4.1 Related Work

To begin with, we provide a brief overview of reinforcement learning, related to our work.

Our ultimate goal is to develop a policy that maximizes information gain not just after executing a single action (akin to teleporting the robot from point A to point B), but rather after executing a sequence of actions. Each action in this sequence represents a discrete movement along the entire path from A to B.

That is a typical Reinforcement Learning problem: an agent interacts with its environment and learns from a feedback signal. At each time step t, the agent executes an action A_t, receives observation O_t, receives reward R_t. The environment receives the action A_t, emits observation O_{t+1}, emits reward R_{t+1}, increments t. The goal is to select actions to maximize total future reward. The prediction of the future rewards is called the value function, and is noted V_t. The interested reader can find a comprehensive introduction to Reinforcement Learning in [71].

MuZero [66] is a really efficient Reinforcement Learning method designed to play games without knowing the rules, but learning them from the interactions. The novelty in MuZero lies in the introduction of a hidden state, denoted S. Instead of solely interacting with the environment, the model transforms the observations into a hidden state. This hidden state is then updated iteratively by a recurrent process that receives the previous hidden state and a hypothetical next action. In the MDP described above, this hidden state is used as the state. This hidden state allows the agent to represent the state in the way that is the most relevant to the task. Building upon this, the search is done over hypothetical future trajectories, through a Monte Carlo Tree Search (MCTS) [25], that outputs a recommended policy π_t and value V_t. The next action, A_{t+1}, is selected applying π_t. The environment receives A_{t+1}, generates a new observation O_{t_+1} and a new reward, R_{t+1}. The agent / environment interaction constitutes an episode, which is stored in a replay buffer. The model is then trained on trajectories sampled from the replay buffer to jointly learn three functions:

- prediction: $f : S_t \rightarrow \pi_t, V_t$,
- dynamics: $g : S_t, A_t \rightarrow R_{t+1}, S_{t+1}$,
- representation: $h : O_t \rightarrow S_t$.

Finally, let us note that in MuZero, an observation is the sequence of last images and actions. The images are directly the images from the game, and the actions are the possible moves in the game.

6.4.2 Preliminary work

We have already undertaken initial work to facilitate the learning of our task. Specifically, we have established a framework that enabled the training of a MuZero agent in a simple exploration task. The framework has three key components:

- a ROS simulation environment,
- the MuZero agent itself,
- SEED-RL, an architecture designed to facilitate the efficient training of MuZero.

SEED-RL is a framework consisting of a single learner and multiple actors. The active interactions between the agent and the environment, and the search in the Monte Carlo Tree Search (MCTS), are performed by the actors, which can be distributed across multiple machines. On the other hand, training occurs in a centralized manner on a single learner.

In our preliminary work, we simplified the world into a 2D grid devoid of obstacles. This world is represented as an image, where pixels are categorized as black (representing unknown space), white (representing discovered space), or red (indicating the current robot's position). The Lidar field of view is depicted as a white disk centered on the robot. As the robot navigates through the world, it progressively discovers more space, increasing the number of white cells in the image. The action space is also simplified, with the agent having a choice of five actions: remain stationary, move right, move left, move up, or move down.

In this simplified simulation, the reward is determined by the number of newly discovered cells, and it becomes negative if the robot remains stationary. The observation consists of a history of previous images and actions. With this simplified setup, we successfully validated our framework and trained a MuZero agent, as illustrated in Figure 6.3.

6.4.3 Methodology Concept

This section outlines our methodology concept for transitioning from our current simplified simulation and task to a coverage task in a more realistic and complex simulation environment. Ultimately, this transition leads to the central topic of this research: an exploration policy designed to enhance map quality. As mentioned earlier, for MuZero, an observation is precisely defined as the sequence of the last images and actions. For simplicity, when we use the term *observation* in what follows, we are specifically referring to the last image in this sequence.

Simulation Firstly, our objective is to validate our framework using a more intricate simulation setup. Our plan involves utilizing Isaac Sim as the simulation platform, where a Husky robot will navigate a world containing obstacles. Isaac-Sim [12] is a simulator provided by Nvidia, known for its computational efficiency, capable of simulating very large scenes with thousands of assets and robots, whereas Gazebo is more limited.

[12] `https://developer.nvidia.com/isaac-sim`

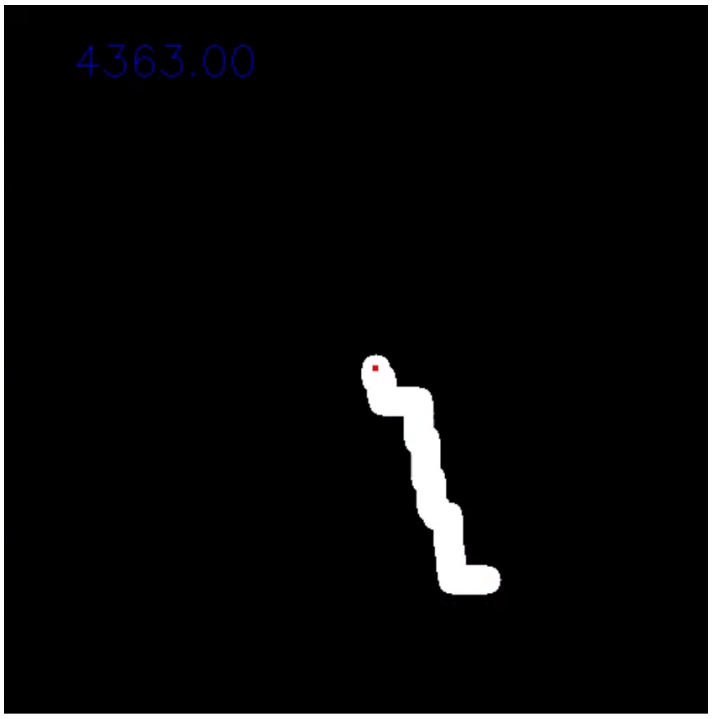

Figure 6.3: Visualization of the simple simulation with a MuZero agent exploring space. Red dot: the robot, black: unknown space, white: discovered space. Top left: total reward.

Coverage Task The initial step of our methodology concept involves training a MuZero agent to perform a more complex task than what was conducted in the basic simulation used in the preliminary work. This task consists in exploring the environment while avoiding obstacles. To enable this, we will modify the observations, reward, and action space. The observation will be a ternary occupancy grid, encoding unknown space (grey), occupied space (black) and free space (white), with the robot's current position in red. This observation will be built from the occupancy grid constructed from the robot's interactions in the environment, with traditional ROS packages as previously done in this research, with expanded obstacles. The reward will remain based on the number of newly discovered cells, with a small penalty when the robot remains stationary and a huge penalty when it collides with an obstacle. The action space will be expanded by including diagonal moves, resulting in a set of nine possible actions: remaining stationary, or moving to one of the eight adjacent cells. When constructing this task, we will consider two methods for controlling the robot. Although actions have been defined, the process of translating these actions into actual robot movement has not been specified. Specifically, the choice of action determines which adjacent cell's center the robot must move to. We will now consider how the robot will execute this movement. The first "control" approach will consist in regularly sampling the segment from the current robot position to the center of the target cell, and teleporting the robot to these sampled points. The

second approach will utilize a conventional planner and controller, chosen for their reliability and execution speed. The advantage of the first method is its speed and robustness, two important factors when training a reinforcement learning agent for hundreds of thousands of steps. On the other hand, the second method is more easily transferable to a real robot.

Map Quality Exploration Task The second step is to train an agent on an even more complex task, which is the core challenge of our research: exploring while considering map quality. Building upon the coverage task, we will maintain the same action space while modifying the observations and the reward. For this task, we will utilize a predictor model similar to the one presented in Chapter 5. This model will predict a 2d-grid representing map quality based on the ground slice of the 3d-grid of viewpoint statistics. From this 2d-grid, we will generate the observation, which will be a 3-channel image comprising:

1. the heatmap corresponding to the normalized 2d-grid quality,
2. the occupancy grid with expanded obstacles,
3. and the position of the robot.

Additionally, we will use the same heatmap to compute the reward. Instead of providing reward based on newly discovered space after each action, the reward will be a function of the increment in the total value of the heatmap following each action. We will keep a small penalty for not moving and a high penalty for colliding with obstacles. However, we acknowledge a potential risk in this approach, as it heavily relies on the map quality predictor, which may be too noisy for the task. A possible mitigation could be to increase the number of channels in the image, and to provide separate heatmaps for each viewpoint statistic on distinct channels instead of using the predicted 2d-grid quality. With this approach, the noise in the observation should be more limited. Then, only the overall quality prediction would be employed to calculate the reward. This prediction is expected to be more robust and better suited for reward computation.

6.4.4 Experimental plan project

Once the methodology is implemented, we plan on evaluating our agents against baselines. For the coverage task, the agent will be compared to state-of-the-art methods, such as [13] introduced earlier. Then, for the map quality exploration task, the agent will be compared to a new NBV policy baseline, based on viewpoint statistics, developed using the work presented in the previous section. We are considering the selection of the NBV with an RRT* planner [39], searching this time for the next-best-trajectory instead of simply performing a

one-step-look-ahead prediction. In both cases, we plan on performing the evaluation on quality vs discovered space curves, as done earlier in this chapter, along with a comparison of the execution speed.

6.5 Summary

In this chapter, we developed a methodology for Next-Best-View (NBV) selection based on observation viewpoint statistics. This work leveraged the observation viewpoint statistics 3D grid presented in Chapter 5. In this work we created four policies, each for a specific statistic (number of observation n_{obs}, minimum range r_{min}, number of angular sectors n_{Ω} or spherical variance σ_{θ}).

Our method involved calculating a costmap and identifying candidate NBVs. Following that, the policy consisted in a balance between the cost and the expected information gain of the candidate NBVs. The expected information gain of each candidate was calculated from the information gain for each cuboid region, that was defined as the surprisal.

Our results were depicted through quality vs. discovered space curves, in environments differently challenging with respect to the reconstruction, ranging from most structured to most unstructured. We compared our methods against different baselines: random-frontier, closest-frontier, and random-free. Generally, our NBV policies produced better map than baselines during the exploration process. Notably, the improvement in map quality was most pronounced in challenging environments.

As we look ahead, we propose a methodology concept to solve this task using deep Reinforcement Learning, where the observations and rewards are derived from the work presented in the preceding chapters of this thesis.

7

Conclusion

The motivation behind this thesis was to address the challenge of an autonomous ground robot exploring an environment, constructing a map from its 3D-lidar observations, all with the overarching objective of enhancing map quality in the process.

During our journey, we discovered that some concepts that may have seemed trivial at first glance were, in fact, challenging tasks in natural environments. These environments are simultaneously unstructured and sparse, two difficulties that amplify each other when building maps. Consequently, the first challenge we encountered was related to map quality. Defining what constitutes a good map is a complex question that leads to an entire chapter in this thesis.

With the first question answered, we could focus on the core motivation behind this research: how to integrate map quality criteria into exploration in a natural environment. Our intuition was that local map quality was closely tied to how the local area was observed. To explore this further, we developed statistics summarizing how the area was observed, our observation viewpoint statistics. We demonstrated that these statistics were indeed indicators of local map quality. As a result, when local map quality can be estimated a priori, it eliminates the need for a reference map, which is typically unavailable in exploration tasks.

Building on this insight, we incorporated these indicators into Next-Best-View exploration policies, where the information gain is computed based on these statistics. Exploring with these policies led to better maps than traditional exploration policies.

7.1 Summary

7.1.1 Local Approach and Map Quality

The initial phase of our work involved establishing a framework to enable the evaluation of methods we would later develop, all with the primary objective of creating an exploration strategy for potentially vast natural environments, aiming to enhance map quality. As we delved deeper into this research, it became clear that adopting a localized perspective was necessary. This localized approach not only enabled our exploration strategy but also its evaluation. To address this need, we decided to divide the environment into smaller cuboid regions, each containing local information, such as map quality and viewpoint statistics.

Chapter 3 framework: local maps and experiments formulated our methodology for this localized approach. In this chapter, we detailed how we extracted local regions, specifically intersecting cuboid regions, from both the map built from the robot's 3D-lidar observations, expressed as a 3D-grid, and the reference map, another 3D-grid. These cuboid regions formed the foundation upon which we built all of our work. Chapter 3 also introduced our

experimental framework, which encompassed both simulations and real-world experiments. It explained how we built the maps from the 3D-lidar data using Octomap and transformed it into a 3D-grid of occupancy likelihoods. Similarly, it described how we transformed the reference map, either the mesh from the simulation or the point cloud from the Total Station in the real-world experiments, into a comparable 3D-grid. Finally, it outlined how we generated simulated environments, featuring different assets characterized by varying levels of complexity with respect to the reconstruction. Additionally, it introduced the various types of experiments we considered throughout this research.

When the aim of an exploration strategy was to enhance map quality, as it was in this research, the first step was to evaluate the map quality itself. However, in the context of natural environments, assessing 3D map quality posed significant challenges. Traditional map quality assessment usually sought a single metric to summarize the overall map quality. Nevertheless, in our case, it was necessary to obtain a localized measure of map quality. Firstly, because map quality varied within a potentially large and complex environment. Secondly, because this localized approach enabled the implementation and evaluation of our exploration policies. Hence, our work, presented in **Chapter 4 measuring map quality**, utilized the cuboid regions introduced previously.

In the scenario we explored, the robot used 3D-Lidar observations to construct a 3D-grid map with occupancy likelihood information for each voxel. In this context, commonly used map quality measures, such as surface coverage and reconstruction accuracy, might not always provide meaningful information. This is especially true in natural environments, which are both unstructured and sparse. Mapping in such environments is further complicated by the challenges related to the robot's localization within the map and by the increased noise level in the 3D-lidar inherent to natural environments. In **Chapter 4**, we demonstrated the limitations of traditional metrics, initially in a simulation, and later in a real-world experiment. We investigated various 3D reconstruction metrics, including both conventional and less conventional ones, with the specific goal of reliably measuring map quality. Leveraging the cuboid regions introduced before, in **Chapter 4**, we computed six distinct metrics: surface coverage, reconstruction accuracy, Average Hausdorff Distance, Cohen's Kappa coefficient, Kullback-Leibler Divergence, and Wasserstein Distance to measure map quality at a local scale. Then, we developed a dedicated methodology for evaluating these metrics, involving controlled 3D reconstructions by progressively degrading ground-truth cuboids using six distinct degradation models. Furthermore, we proposed a method to assess the metrics' performance as classifiers with two distinct quality levels. Our evaluation showed that no single metric could provide a meaningful measure in all situations. The choice of metric largely depended on the specific purpose of the quality assessment. For the topic of interest

in this research, autonomous robot mapping in natural environments, the Average Hausdorff Distance appeared to be the most suitable metric, particularly because of its robustness.

7.1.2 Linking Map Quality to Viewpoint Statistics and Integrating them into Next- Best-View

Obtaining a robust local measure of map quality allowed us to delve into the central question of this research: how to explore a natural environment while incorporating map quality criteria? To address this, we proposed estimating map quality a priori by relying on viewpoint statistics derived directly from the robot's data. Building upon that, we integrated this prior, extracted from the viewpoint statistics, into an exploration policy through the selection of the Next-Best-View.

Chapter 5 Linking Map Quality and View-Point Statistics outlined our methodology. Four viewpoint statistics were introduced: the number of observations, their minimum range, the number of angular sectors covered by a viewpoint, and the spherical variance of the viewpoints. We detailed their computation in a 3D-grid, the same 3D-grid used in **Chapter 4** to assess local map quality. Our findings highlighted the link between map quality and viewpoint statistics through quality vs. statistics plots. More importantly, we rigorously proved through hypothesis testing that these viewpoint statistics were valid indicators of local map quality. Finally, we trained a Random Forest Regressor to model quality from the viewpoint statistics, and we evaluated the relative importance of each feature in the learning.

Notably, in **Chapter 5**, our results emphasized the importance of different statistics in structured or unstructured environments. In unstructured environments, a combination of statistics, including the number of observations, the minimum range of the observations, and one reflecting viewpoint diversity, such as the number of angular sectors of the viewpoints or their spherical variance, proved essential for accurate prediction, contrasting with structured environments where the number of observations alone might suffice.

The contributions from **Chapter 5** laid the groundwork for the exploration policies discussed in **Chapter 6 Autonomous Exploration with View-Point Statistics-Based Policy**.

Chapter 6 introduced exploration policies based on these viewpoint statistics individually. Once we identified candidates for the Next-Best-View, we calculated the expected information gain of each candidate. This method built upon the 3D grid of viewpoint statistics, where we compared the current and updated 3D grid, considering the new viewpoints from the candidate. The expected information gain for the candidate was defined as the sum of the surprisal in each cell. Finally, the Next-Best-View was selected as the candidate that best balanced high information gain and low distance.

The results were presented in **Chapter 6**, in environments differently challenging with respect to the reconstruction. Our policies were compared against traditional baselines, such as random or closest frontier explorations. These results demonstrated that our Next-Best-View policies, based on viewpoint statistics, produced better maps than traditional baselines during the exploration process. Notably, the improvement in map quality was most pronounced in challenging environments.

Chapter 6 presented our conceptual methodology to enhance these policies using reinforcement learning, where observations and rewards were derived from the groundwork laid out in this thesis. This concept would help scale our policies, enabling the selection of the Next-Best-View along the complete trajectory.

7.2 Future Work

This section summarizes the questions raised by this thesis that will be investigating in future work.

Map quality Firstly, Chapter 4 showed that the Average Hausdorff Distance appeared the more robust metric when measuring map quality in natural environments. Nonetheless, such a metric only considers occupied points, i.e., voxels from the map whose occupancy likelihood is above some selected threshold. It totally discards any information below this threshold. A solution for an even more robust map quality estimation would be to combine metrics, such as the Average Hausdorff Distance and the Kullback-Leibler Divergence, the first being sensitive to Euclidean distance in the errors, the seconds to the level of certainty in the complete cuboid. Secondly, complementary investigation could be done on the Wasserstein Distance. The work presented in Chapter 4 focused on conventional optimal transport, yet it would be interesting in future work to explore unbalanced optimal transport [22]. This would enable measures between positive values that are not necessary probability distributions. That would eliminate the current normalization step, allowing the comparison of cuboids with different mass. This could lead to a more robust map quality assessment.

Predicting Map Quality from Viewpoint Statistics Building upon the work presented in Chapter 5, future work will focus on the learning task. Particularly, it will require acquiring more data to enable training a more robust predictor. Different approaches may be explored. It could be a Random Forest Regressor, as it was the case in this work, but it may also be a Neural Network. With enough data and a Convolutional Neural Network, we could be

able to learn the 3d-grid map quality from the 3d-grids of viewpoint statistics, allowing the incorporation of neighboring cues. It is likely that map quality follows a spatial pattern.

Exploration Policies Finally, following Chapter 6, future work will focus on crafting exploration policies with reinforcement learning. This work will start with implementing the methodology concept described in Chapter 6, and continue with the proposed evaluation plan.

A

Theoretical Metrics Comparison

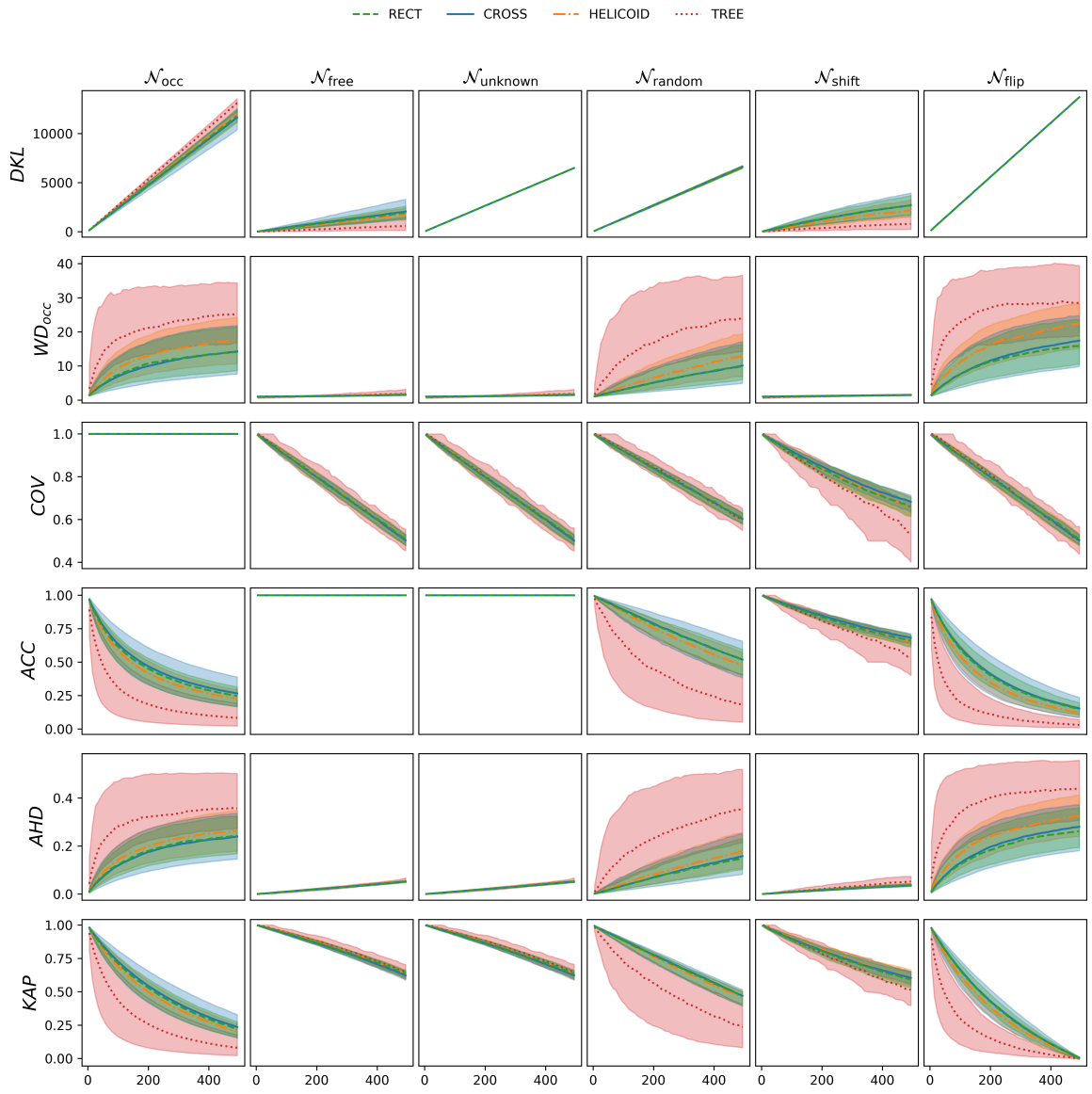

Figure A.1: Illustration of the different metrics behavior when the reconstruction moves further from the ground-truth. One line per metric, one column per type of degradation applied to the gt. For each sub-figure, the x-axis is the number of iteration n, the y-axis the value of the metric θ. The results are grouped by type of asset in the world. In each sub-figure, the line corresponds to the median value from all cuboids in the type of world and the filled area shows the spread of 80% of the population.

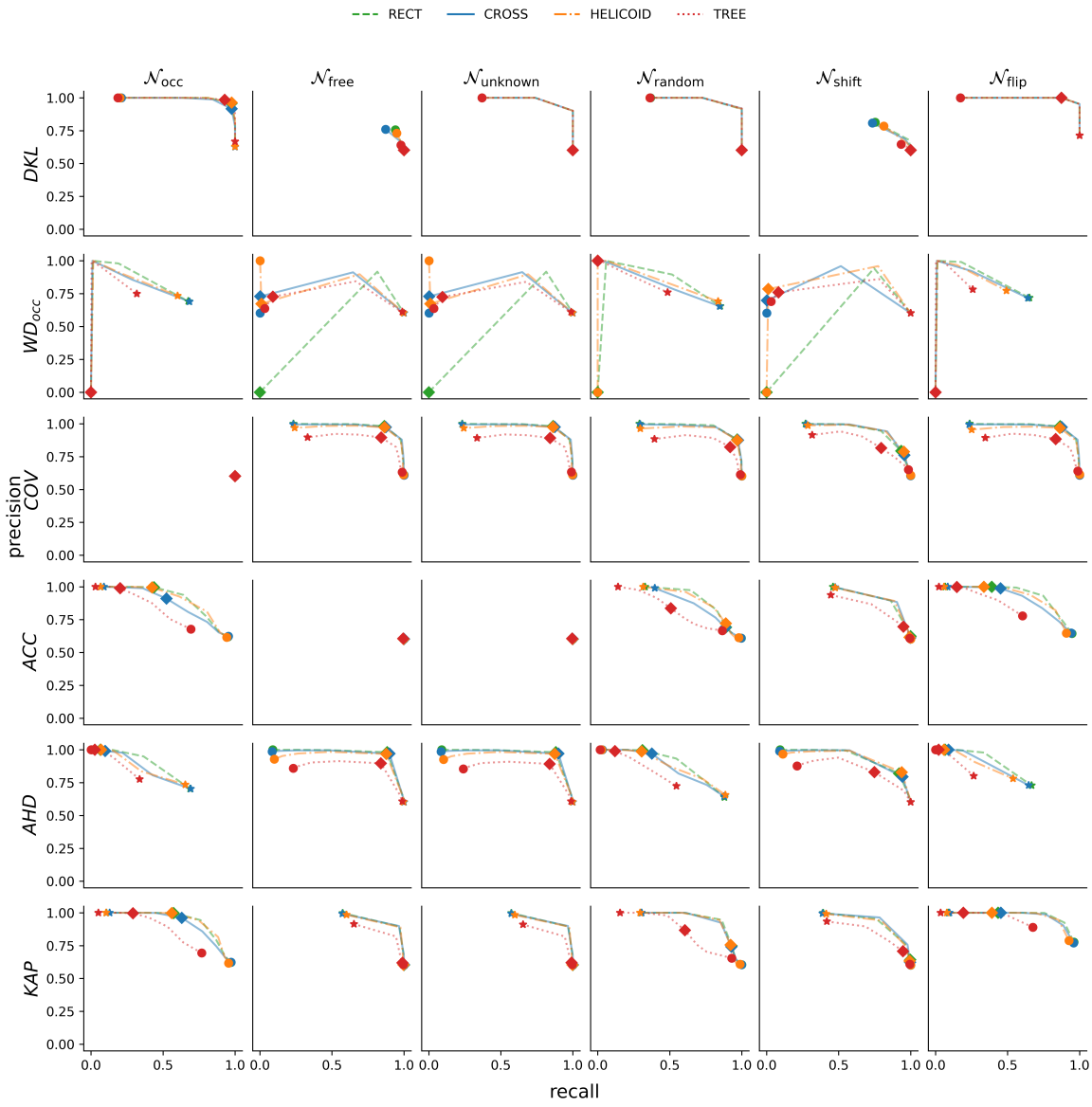

Figure A.2: Precision-Recall curves when we vary the value of the threshold $\hat{\theta}$ for each metric. One line per metric, one column per type of degradation, one color per type of world. The three markers display precision-recall points for three values of $\hat{\theta}$ for each metric. The results are displayed for a level of degradation of the cuboid of 20%. The points in (0,0) corresponds to points where it is not possible to compute precision and recall (division by 0 in Eq. 4.13 and 4.14)

Glossary

UAV: Unmanned Aerial Vehicule
UGV: Unmanned Ground Vehicule
IMU: Inertial Measurement Unit
GPS: Global Positioning System
RTK-GPS: Real Time Kinematics-GPS
Lidar: LIght Detection And Ranging
SLAM: Simultaneous Localization and Mapping
NBV: Next-Best-View
AHD: Average Hausdorf Distance
COV: Surface Coverage
ACC: Reconstruction Accuracy
WD: Wasserstein Distance
DKL: Kullback-Leibler Divergence
KAP: Cohen's Kappa
$\mathcal{U}_{occ}$: Set of cuboids in occupied space
$\mathcal{U}_{empty}$: Set of cuboids in empty space
C_{gt}: A cuboid from the ground-truth dataset
C_{rec}: A cuboid from the reconstruction dataset
n_{obs}: Number of observations
r_{min}: Minimun range of the observations
n_{Ω}: Number of angular sectors covered by the observations
σ_{θ}: Spherical variance of the observations
RECT: Experiment in a world with rectangular cuboids
CROSS: Experiment in a world with cross extruded shape
HELICOID: Experiment in a world with helicoidal shapes
TREES: Experiment in a world with simulated trees
REAL: Real world experiment
NAVIGATION: Experiments where the robot navigates autonomously towards a goal

TELEPORT: Experiments where the robot is teleported on a goal
FRONT: Experiment where the robot drives toward the object
LAT: Experiment where the robot drives with the object on the side
WAYPOINTS: Experiments where the robot follows autonomously a list of waypoints
RANDOM: Experiments where the robot is teleported to a goal that is randomly sampled
NBV-XP: Experiments where the robot is teleported to a goal selected by the policy

List of Figures

Bibliography

[1] P. F. Alcantarilla, C. Beall, and F. Dellaert. Large-Scale Dense 3D Reconstruction from Stereo Imagery. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[2] R. Almadhoun, A. Abduldayem, T. Taha, L. Seneviratne, and Y. Zweiri. Guided next best view for 3D reconstruction of large complex structures. *Remote Sensing*, 11(20):1–20, 2019.

[3] S. Aravecchia, M. Clausel, and C. Pradalier. Comparing Metrics for Evaluating 3D Map Quality in Natural Environment. *under review*, 2023.

[4] S. Aravecchia, A. Richard, M. Clausel, Pradalier, and Cédric. Next-Best-View selection with view-points statistics on the observations. In *IEEE International Conference on Intelligent Robots and Systems*, 2023.

[5] S. Aravecchia, A. Richard, M. Clausel, and C. Pradalier. Measuring 3D-reconstruction quality in probabilistic volumetric maps with the Wasserstein Distanc. In *International Symposium on Robotics (ISR Europe)*, 2023.

[6] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. MESH: Measuring errors between surfaces using the Hausdorff distance. In *Proceedings - 2002 IEEE International Conference on Multimedia and Expo, ICME 2002*, volume 1, 2002.

[7] P. Babin, P. Dandurand, V. Kubelka, P. Giguère, and F. Pomerleau. Large-Scale 3D Mapping of Subarctic Forests. *Springer Proceedings in Advanced Robotics*, 16:261–275, 2021.

[8] L. Bartolomei, L. Teixeira, and M. Chli. Perception-aware Path Planning for UAVs using Semantic Segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5808–5815, 2020.

[9] A. Batinovic, T. Petrovic, A. Ivanovic, F. Petric, and S. Bogdan. A Multi-Resolution Frontier-Based Planner for Autonomous 3D Exploration. *IEEE Robotics and Automation Letters*, 6(3):4528–4535, 2021.

[10] I. Ben Salah, S. Kramm, C. Demonceaux, and P. Vasseur. Summarizing large scale 3D mesh for urban navigation. *Robotics and Autonomous Systems*, 152:104037, 2022.

[11] B. Berger, M. S. Waterman, and Y. W. Yu. Levenshtein Distance, Sequence Comparison and Biological Database Search. *IEEE Transactions on Information Theory*, 67(6):3287–3294, 2021.

[12] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[13] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon next-best-view planner for 3D exploration. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1462–1468, 2016.

[14] L. Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001.

[15] L. Breiman. Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*, 58:3–42, 2002.

[16] A. Bulbul, T. Capin, G. Lavoue, and M. Preda. Assessing visual quality of 3-D polygonal models. *IEEE Signal Processing Magazine*, 28(6):80–90, 2011.

[17] S. T. Burgard, Wolfram Mark, Moors Dieter, Fox Reid and Sebastian. Collaborative Multi-Robot Exploration. In *IEEE international conference on robotics and automation*, 2000.

[18] N. O. Center and A. A. N. C. Services. Lidar 101 : An Introduction to Lidar Technology , Data , and Applications. *NOAA Coastal Services Center*, (November):76, 2012.

[19] G. Chahine, C. Pradalier, G. Chahine, C. Pradalier, S.-a. Alignment, and N. Outdoor. Semantic-aware spatio-temporal Alignment of Natural Outdoor Surveys To cite this version : HAL Id : hal-03738518 Semantic-aware spatio-temporal Alignment of Natural Outdoor Surveys. 2022.

[20] G. Chahine, M. Vaidis, F. Pomerleau, and C. Pradalier. Mapping in unstructured natural environment: a sensor fusion framework for wearable sensor suites. *SN Applied Sciences*, 3(5):1–14, 2021.

[21] S. W. Cheng, T. K. Dey, and J. R. Shewchuk. Delaunay mesh generation. *Delaunay Mesh Generation*, pages 1–386, 2012.

[22] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard. Unbalanced optimal transport: Dynamic and Kantorovich formulations. *Journal of Functional Analysis*, 2018.

[23] J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[24] P. I. Corke, W. Jachimczyk, and R. Pillat. *Robotics, vision and control: fundamental algorithms in MATLAB*. Springer, 2011.

[25] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[26] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems*, pages 1–9, 2013.

[27] M. P. Fay and M. A. Proschan. Wilcoxon-Mann-Whitney or T-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics Surveys*, 4:1–39, 2010.

[28] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. POT python optimal transport library. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[29] D. Fox, S. Thrun, and W. Burgard. *Probabilistic Robotics*. Kybernetes, 2006.

[30] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[31] J. Gonzalez, A. Ollero, and A. Reina. Map building for a mobile robot equipped with a 2D laser rangefinder. *Proceedings - IEEE International Conference on Robotics and Automation*, (pt 3):1904–1909, 1994.

[32] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. *IEEE International Conference on Robotics and Automation*, 2005:2432–2437, 2005.

[33] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream To Control: Learning Behaviors By Latent Imagination. *8th International Conference on Learning Representations, ICLR 2020*, pages 1–20, 2020.

[34] G. Hardouin, J. Moras, F. Morbidi, J. Marzat, and E. M. Mouaddib. Next-Best-View planning for surface reconstruction of large-scale 3D environments with multiple UAVs. *IEEE International Conference on Intelligent Robots and Systems*, pages 1567–1574, 2020.

[35] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[36] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza. An information gain formulation for active volumetric 3D reconstruction. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:3477–3484, 2016.

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

[38] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[39] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 7(30):846–894, 2011.

[40] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Eurographics symposium on Geometry processing*, volume 7, 2006.

[41] A. Khoche, M. K. Wozniak, D. Duberg, and P. Jensfelt. Semantic 3D Grid Maps for Autonomous Driving. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2022-Octob:2681–2688, 2022.

[42] I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.

[43] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa. Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects. *Journal of Real-Time Image Processing*, 10(4):611–631, 2015.

[44] M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, R. Siegwart, M. Hutter, and K. Alexis. Autonomous Teamed Exploration of Subterranean Environments using Legged and Aerial Robots. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3306–3313, 2022.

[45] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1), 1951.

[46] J. Laconte, S. P. Deschênes, M. Labussière, and F. Pomerleau. Lidar measurement bias estimation via return waveform modelling in a context of 3D mapping. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:8100–8106, 2019.

[47] S. LaValle and A. Others. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, 2001.

[48] G. Louppe. *Understanding Random Forests: From Theory to Practice*. PhD thesis, 2014.

[49] A. Mahé, A. Richard, S. Aravecchia, M. Geist, and C. Pradalier. Evaluation of prioritized deep system identification on a path following task. *Journal of Intelligent and Robotic Systems*, 2021.

[50] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics,*, 18(1):50–60, 1947.

[51] N. A. Massios and R. B. Fisher. A Best Next View Selection Algorithm Incorporating a Quality Criterion. *D*, 2, 1998.

[52] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *IEEE international conference on robotics and automation*, pages 116–121, 1985.

[53] D. Müller, I. Soto-Rey, and F. Kramer. Towards a guideline for evaluation metrics in medical image segmentation. *BMC Research Notes*, 15(1):1–7, 2022.

[54] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:1366–1373, 2017.

[55] C. Oliveira, S. Aravecchia, C. Pradalier, V. Robin, and S. Devin. The use of remote sensing tools for accurate charcoal kilns' inventory and distribution analysis: Comparative assessment and prospective. *International Journal of Applied Earth Observation and Geoinformation*, 105:102641, dec 2021.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

[57] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[58] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto. An informative path planning framework for UAV-based terrain monitoring. *Autonomous Robots*, 44(6):889–911, 2020.

[59] M. L. Puterman. Markov decision processes: discrete stochastic dynamic programming. *John Wiley & Sons*, 2014.

[60] A. Richard, S. Aravecchia, M. Geist, and C. Pradalier. Learning Behaviors through Physics-driven Latent Imagination. In *Conference on Robot Learning*, pages 1190–1199, 2021.

[61] A. Richard, S. Aravecchia, T. Schillaci, M. Geist, and C. Pradalier. How To Train Your HERON. *IEEE Robotics and Automation Letters*, 2021.

[62] J. Ruckin, L. Jin, and M. Popovic. Adaptive Informative Path Planning Using Deep Reinforcement Learning for UAV-based Active Sensing. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4473–4479, 2022.

[63] W. J. Rucklidge. Efficiently Locating Objects Using the Hausdorff Distance. *International Journal of Computer Vision*, 24(3):251–270, 1997.

[64] J. Santos, M. Oliveira, R. Arrais, and G. Veiga. Autonomous scene exploration for robotics: A conditional random view-sampling and evaluation using a voxel-sorting mechanism for efficient ray casting. *Sensors (Switzerland)*, 20(15):1–30, 2020.

[65] R. Saravanan and R. D. Levine. Surprisal analysis of diffusion processes. *Chemical Physics*, 556(November 2021):111450, 2022.

[66] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[67] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to autonomous mobile robots*. 2011.

[68] S. Song and S. Jo. Surface-Based Exploration for Autonomous 3D Modeling. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4319–4326, 2018.

[69] S. Soudarissanane, R. Lindenbergh, M. Menenti, and P. Teunissen. Scanning geometry: Influencing factor on the quality of terrestrial laser scanning points. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(4):389–399, 2011.

[70] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. *Robotics: Science and Systems*, 1:65–72, 2005.

[71] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. 2018.

[72] A. A. Taha and A. Hanbury. Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool. *BMC Medical Imaging*, 15(1), 2015.

[73] D. E. Tyler. Statistical Analysis for the Angular Central Gaussian Distribution on the Sphere. *Biometrika*, 74(3):579–589, jan 1987.

[74] C. Villani. *Optimal transport: old and new*. Springer Verlag., 2009.

[75] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G. L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz,

M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020.

[76] G. Vosselman. *Airborne and Terrestrial Laser Scanning*. Whittles Publishing, 2010.

[77] Y. Wang and A. Del Bue. Where to Explore Next? ExHistCNN for History-Aware Autonomous 3D Exploration. In *Computer Vision–ECCV European Conference*, pages 125–140. Springer International Publishing, 2020.

[78] U. Wickramasinghe, E. Remelli, G. Knott, and P. Fua. Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12264 LNCS:299–308, 2020.

[79] T. Wiemann, F. Igelbrink, S. Pütz, and J. Hertzberg. A File Structure and Reference Data Set for High Resolution Hyperspectral 3D Point Clouds. *IFAC-PapersOnLine*, 52(8):93–98, 2019.

[80] J. Williams, S. Jiang, M. O'brien, G. Wagner, E. Hernandez, M. Cox, A. Pitt, R. Arkin, and N. Hudson. Online 3D Frontier-Based UGV and UAV Exploration Using Direct Point Cloud Visibility. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2020-Septe:263–270, 2020.

[81] B. Yamauchi. Frontier-based approach for autonomous exploration. *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA*, pages 146–151, 1997.

[82] B. Zhou, H. Xu, and S. Shen. RACER: Rapid Collaborative Exploration With a Decentralized Multi-UAV System. *IEEE Transactions on Robotics*, pages 1–19, 2023.

[83] J. Zhou, X. Fu, L. Schumacher, and J. Zhou. Evaluating geometric measurement accuracy based on 3d reconstruction of automated imagery in a greenhouse. *Sensors (Switzerland)*, 18(7):1–16, 2018.

[84] H. Zhu, J. J. Chung, N. R. J. Lawrance, R. Siegwart, and J. Alonso-Mora. Online Informative Path Planning for Active Information Gathering of a 3D Surface. In *IEEE International Conference on Robotics and Automation*, 2021.