DISS. ETH NO. 30211

MULTI-RESOLUTION FOR EFFICIENT, SCALABLE AND ACCURATE VOLUMETRIC MAPPING AND PLANNING IN ROBOTICS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES

(Dr. sc. ETH Zurich)

presented by

VICTOR JOHAN FREERK REIJGWART

MSc. ETH in Robotics, Systems and Control

born on 22.01.1994

accepted on the recommendation of

Prof. Dr. Roland Siegwart
Prof. Dr. Stefan Leutenegger
Prof. Dr. Cyrill Stachniss

2024

Autonomous Systems Lab
Institute for Robotics and Intelligent Systems
ETH Zurich
Switzerland

Abstract

As robots evolve beyond industrial settings to address broader challenges, such as autonomous inspection, home assistance, and search and rescue, there is a growing demand for them to autonomously navigate and perform meaningful tasks in increasingly large, unstructured, and unknown environments. Despite improvements in hardware, sensing, and computational technologies enabling greater robot agility and perception, a significant bottleneck remains in their software, particularly in autonomous mapping and navigation capabilities. Volumetric maps offer a general, safe, and task-agnostic representation of the environment but are hindered by their excessive computational and memory demands, limiting their practical use on small and affordable robots.

This doctoral thesis investigates the use of adaptive representations as a solution to these challenges, focusing on enhancing the scalability, efficiency, and accuracy of volumetric maps. Recognizing that the value of volumetric maps is determined by the benefits they bring to downstream tasks, we study local and global planning as two representative applications. Leveraging hierarchical, multi-resolution approaches, this work aims to dynamically balance the trade-off between detail and computational cost, tailored to the mission's needs.

The main contribution of this thesis is the development of a mathematically rigorous multi-resolution mapping framework, named *wavemap*, that adjusts the map's resolution based on the environment's geometry without reliance on heuristics. The Multi-Resolution Analysis (MRA) theory guarantees that using wavelet decomposition, new observations can safely and efficiently be integrated into the map in a coarse-to-fine manner. The resulting gains in computational efficiency, together with early stopping criteria for the integrator, allow us to use a more complex measurement model that improves the capture of thin objects, thereby enhancing the safety and reliability of robotic operations. The framework is extensively evaluated on synthetic and real data, and shown to efficiently reconstruct large-scale environments while accurately capturing fine details. Beyond significant improvements in terms of scalability and map quality, the framework's flexibility facilitates its use across a wide range of sensors and applications.

Our second and third contributions are efficient methods for reactive obstacle avoidance and deterministic global path planning, utilizing hierarchical representations and algorithms alongside the *wavemap* framework to enable rapid, reliable navigation through complex environments. Experimental evaluations on maps of diverse, real environments and deployments on a micro aerial vehicle demonstrate the superiority of these approaches over existing methods in terms of efficiency, accuracy, and flexibility, underscoring their potential to significantly advance the field of robotic mapping and navigation.

In sum, this doctoral thesis presents a comprehensive solution to the challenges of volumetric mapping and planning in robotics, paving the way for more autonomous, efficient, and versatile robotic systems capable of operating in diverse and changing environments.

Zusammenfassung

Während sich Roboter über industrielle Anwendungen hinaus entwickeln, um breitere Herausforderungen wie autonome Inspektionen, Hilfe im Haushalt und Such- und Rettungsaktionen anzugehen, wächst die Nachfrage nach ihrer Fähigkeit, autonom in zunehmend grossen, unstrukturierten und unbekannten Umgebungen zu navigieren und sinnvolle Aufgaben zu erfüllen. Trotz Verbesserungen in der Hardware, Sensorik und Computertechnologie, die eine grössere Agilität und Wahrnehmungsfähigkeit der Roboter ermöglichen, bleibt ein signifikanter Engpass in ihrer Software, insbesondere bei den Fähigkeiten zur autonomen Kartierung und Navigation. Volumetrische Karten bieten eine allgemeine, sichere und aufgabenagnostische Darstellung der Umgebung, sind jedoch durch ihren übermässigen Rechen- und Speicherbedarf eingeschränkt, was ihren praktischen Einsatz auf kleinen und erschwinglichen Robotern begrenzt.

Diese Doktorarbeit untersucht die Verwendung adaptiver Darstellungen als Lösung für diese Herausforderungen und konzentriert sich auf die Verbesserung der Skalierbarkeit, Effizienz und Genauigkeit volumetrischer Karten. In der Erkenntnis, dass der Wert volumetrischer Karten durch die Vorteile bestimmt wird, die sie für nachgelagerte Aufgaben bringen, untersuchen wir lokale und globale Planung als zwei repräsentative Anwendungen. Durch die Nutzung hierarchischer, mehrstufiger Ansätze zielt diese Arbeit darauf ab, den Kompromiss zwischen Detailgenauigkeit und Rechenkosten dynamisch auszugleichen, angepasst an die Bedürfnisse der Mission.

Der Hauptbeitrag dieser Arbeit ist die Entwicklung eines mathematisch rigorosen Mehr-Ebenen-Mapping-Rahmenwerks, benannt als *wavemap*, das die Kartenauflösung basierend auf der Geometrie der Umgebung ohne Abhängigkeit von Heuristiken anpasst. Die Theorie der Multi-Resolution-Analyse (MRA) garantiert, dass durch die Verwendung der Wavelet-Zerlegung neue Beobachtungen sicher und effizient in die Karte in einer grob-zu-fein Manier integriert werden können. Die daraus resultierenden Gewinne an Recheneffizienz, zusammen mit frühen Stoppkriterien für den Integrator, ermöglichen den Einsatz eines komplexeren Messmodells, das die Erfassung dünner Objekte verbessert und somit die Sicherheit und Zuverlässigkeit von Roboteroperationen erhöht. Das Rahmenwerk wird ausführlich anhand von synthetischen und realen Daten bewertet und zeigt, dass es grossflächige Umgebungen

effizient rekonstruieren kann, während es feine Details genau erfasst. Neben signifikanten Verbesserungen hinsichtlich der Skalierbarkeit und Kartenqualität erleichtert die Flexibilität des Rahmens seinen Einsatz über eine breite Palette von Sensoren und Anwendungen.

Der zweite und dritte Beitrag sind effiziente Methoden zur reaktiven Hindernisvermeidung und zur deterministischen globalen Pfadplanung, die hierarchische Darstellungen und Algorithmen zusammen mit der *wavemap* Methode nutzen, um eine schnelle, zuverlässige Navigation durch komplexe Umgebungen zu ermöglichen. Experimentelle Bewertungen auf Karten von diversen, realen Umgebungen und Einsatzauf einem Mikro-Fluggerät demonstrieren die Überlegenheit dieser Ansätze gegenüber bestehenden Methoden in Bezug auf Effizienz, Genauigkeit und Flexibilität und unterstreichen ihr Potenzial, das Gebiet der robotischen Kartierung und Navigation erheblich voranzubringen.

Zusammenfassend präsentiert diese Doktorarbeit eine umfassende Lösung für die Herausforderungen der volumetrischen Kartierung und Planung in der Robotik und ebnet den Weg für autonomere, effizientere und vielseitigere Robotersysteme, die in der Lage sind, in diversen und sich verändernden Umgebungen zu operieren.

Acknowledgements

This thesis represents not just the culmination of considerable work, but also the joy and enrichment that came with it. Foremost, I extend my deepest gratitude to Professor Roland Siegwart for offering me the opportunity to conduct my doctoral studies at the Autonomous Systems Lab. His enthusiasm, encouragement, and unwavering support have been invaluable to me. The environment at the lab, marked by its openness, passion, and collaborative spirit, has been truly inspiring. I extend my sincere thanks to Professor Stefan Leutenegger and Professor Cyrill Stachniss for their willingness to co-examine this thesis, and for doing so much excellent work in our field.

Deep appreciation goes to my supervisors – Dr. Lionel Ott for his rigorous technical mentorship and continually sharpening my understanding of the field, Dr. Cesar Cadena for his wide-ranging guidance and unwavering optimism, and Dr. Juan Nieto for always challenging me to see the bigger picture. Their support was instrumental in the realization of this work.

I owe a special thanks to Dr. Helen Oleynikova and Dr. Alexander Millane for kindling my interest in research during my Master's thesis and encouraging me on the path toward this Ph.D. Thank you to Dr. Marius Fehr for our time working together on the DARPA SubT Challenge and for sharing so much knowledge with me. Finally, a big thank you to Dr. Michael Pantic for our collaboration and passionate, insightful discussions. Throughout this Ph.D., the multitude of projects were made memorable by the collaboration with colleagues and students alike, including the CERBERUS and PILOTING teams, as well as Christian Lanegger, Lukas Bernreiter, Julian Keller, Andrei Cramariuc, and many others. Each project was a unique learning opportunity, thanks to your contributions.

I sincerely thank our technical and administrative staff – Michael Riner, Cornelia Della Casa, Luciana Borsatti – for allowing our lab to run so smoothly.

Finally, my most heartfelt appreciation goes to my friends, family, and wife for the wonderful life we share and for supporting and encouraging me over the years.

June 19, 2024 *Victor Reijgwart*

Funding

The research leading to the publications and results presented in this thesis was supported by the National Center of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation, as well as the Defense Advanced Research Projects Agency (DARPA) under agreement number HR00111820045, and the European Commission under grant number 871542. The presented content and ideas are solely those of the authors.

Contents

Acronyms

SLAM Simultaneous Localization And Mapping

TSDF Truncated Signed Distance Field

ESDF Euclidean Signed Distance Field

MAV Micro Aerial Vehicle

MRA Multi-Resolution Analysis

FWT Fast Wavelet Transform

ROC Receiver Operating Characteristic

AUC Area under the ROC Curve

TPR True Positive Rate

FPR False Positive Rate

RMP Riemannian Motion Policy

EDF Euclidean Distance Field

Chapter 1

Introduction

Robots, long used in industrial settings, are now evolving to address broader global challenges and human welfare needs, taking on diverse tasks like autonomous inspection, home assistance, and search and rescue. Moreover, decreasing hardware costs and advancements in user interfaces make the deployment of commercial and personal service robots socially and economically viable. This new generation of robots, equipped with smaller, more accurate sensors and enhanced computational hardware, is poised to offer greater agility and perception. However, significant gaps in their software, particularly in their autonomy, hinder their ability to reliably perform varied tasks and operate in uncontrolled environments.

A fundamental aspect that limits the flexibility and generalizability of robotic systems is how they model and reason about their surroundings. Mapping has been a key research area within the robotics community over the past decades. Many successful solutions represent the world using sparse sets of distinctive features. While such feature-based maps can scale well to large environments, choosing features is inherently task and environment-dependent. Sparse feature-based methods, therefore, tend to be limited to mapping and localization, or planning in controlled environments.

On the other end of the spectrum, volumetric maps focus on estimating relevant properties of the environment, such as occupancy, at all points in space. They can represent objects of arbitrary shapes and distinguish observed from unobserved space without requiring simplifying assumptions or prior knowledge. This makes volumetric maps particularly safe to deploy in unknown or changing environments. Furthermore, their generality allows a single map to be used for a broad portfolio of tasks. Volumetric

maps allow efficient, typically constant time, retrieval of their estimates at query points. This motivates their frequent use in solving planning problems ranging from collision avoidance and manipulation to autonomous exploration. Beyond their generality in terms of use cases, volumetric maps can be updated using a wide range of sensing modalities – including sonar, radar, LiDAR, and depth cameras – and serve as a good common ground for sensor fusion. Conceptually, this representation is therefore well poised to enable autonomous robots to safely complete more complex tasks in unstructured environments.

Nevertheless, fundamental challenges remain. The computational and memory complexity of storing and integrating new measurements into 3D volumetric maps generally scales linearly with the mapped volume and cubically as a function of the resolution. The latter aspect is particularly problematic since the resolution directly controls the amount of detail a volumetric map can capture and its maximum achievable accuracy. Beyond storing and updating the map, these costs tend to be incurred again in downstream applications with tasks such as obstacle avoidance and collision checking often also scaling cubically in the chosen resolution.

Volumetric methods are a popular choice on research platforms that can afford top-of-the-line computing hardware and where battery life is not a primary concern. However, the cost-to-benefit ratio is still too low to justify their adoption on commercial devices. Notable exceptions, such as high-end drones or surveying solutions, only use volumetric representations for small, local areas at low resolutions or during offline operations.

Motivated by the fact that real environments predominantly consist of free space and the amount of detail needed on surfaces depends on their relevance to the task, adaptive volumetric mapping methods offer a promising avenue for improvement. Multi-resolution representations can, for example, represent uniform areas such as free space at low resolutions, while reconstructing high-resolution details where needed.

In this thesis, we investigate ways to make the resource usage of volumetric maps more scalable. Ideally, we would like the representation to allow a granular trade-off between cost and accuracy that can dynamically be aligned to the mission objectives. Additionally, we examine the impact of this new mapping representation on downstream tasks. Using local and global planning as two representative examples, we show how the advantages of the new representation not only benefit geometric modeling but also reasoning.

In the rest of this section, we will first briefly summarize the author's key takeaways from using existing volumetric mapping frameworks in a number of robotics

competitions and projects. We will then translate them into research objectives to make volumetric mapping systems more useful in practice. Finally, we will briefly summarize the approach behind each contributing thesis chapter.

1.1 Practical considerations

From our experience, volumetric maps work well across indoor, outdoor, structured, and natural environments. We have also successfully used them with sonars, depth cameras, and LiDARs from all major brands. Once a volumetric framework has been configured for a given sensor setup, little to no parameter changes are required to accommodate new environments aside from resolution or other scalability-related adjustments. In the DARPA Subterranean Challenge (SubT), our team, CERBERUS, used volumetric maps for local, global, and exploration planning [1, 2]. More generally, every team in SubT used volumetric maps in at least one part of their system. Some of our team's robots further used voxgraph [3], based on volumetric submaps, as the onboard SLAM solution. In conclusion, volumetric maps offer a good balance between generality and expressiveness. Additionally, although they come with an upfront computational cost, they often reduce the computational effort required downstream and can effectively be reused for multiple tasks.

The most important limitation we experienced is that fixed-resolution volumetric mapping scales poorly. The fact that its memory and computational complexities grow cubically with resolution is a major limitation in practice. It meant that, for CPU-based approaches integrating LiDAR inputs, the best achievable map resolution was limited to around 15cm. GPU-accelerated implementations can overcome computational bottlenecks but run into memory constraints instead. The problem is made worse because the costs in downstream applications often scale similarly to their underlying map. Common operations such as ESDF generation, occupancy-based collision checking, and mesh generation, for example, also scale cubically in the chosen resolution. One common workaround is to use multiple maps, such as a low-resolution map for global mapping, planning, and exploration, complemented by a high-resolution local map for manipulation, traversability estimation, and local planning.

Another important practical issue is that volumetric maps struggle to capture thin objects, such as ropes, poles, fences, and vegetation, which can be hazardous obstacles. One reason for this is that voxel-based representations cannot accurately represent details below their voxel size. Another reason is that most existing measurement models assume that measurement rays are always one voxel thick. On the one hand, when the grid resolution is much higher than the beam density, this leads to aliasing

artifacts. On the other hand, when the resolution is low, this leads to issues such as scanning rays eroding the edges of objects that they miss by less than the width of a voxel.

1.2 Research objectives

The primary goal of this thesis is to enhance the scalability of volumetric mapping. Our research concentrates on multi-resolution methods, which maintain the general applicability and user-friendliness of voxel-based approaches while significantly improving their efficiency and flexibility. We also investigate how these enhancements can be used to bolster accuracy. Moreover, we believe mapping representations should be designed to maximize their utility in downstream applications to ensure their effectiveness and relevance in practice.

To achieve these goals, we define the following core research objectives:

- **Mathematically rigorous multi-resolution**: The resolution should adapt to the environment's geometry without relying on heuristics or hand-tuning.
- **Computational efficiency**: The representation should be both memory and computationally efficient to update, store, and query.
- **Flexibility**: The framework should allow a granular trade-off between accuracy and computational efficiency that can, ideally, dynamically be adjusted to the task.
- **Synergy with applications**: The representation should be easy to use and benefit downstream tasks.

1.3 Approach

The work in this thesis is split into three main parts. In the first part, we investigate how to rigorously, efficiently, and accurately model, store, and update multi-resolution volumetric maps. The second and third parts then focus on how hierarchical volumetric maps can benefit downstream tasks, taking local planning and global planning as two representative and complementary examples. Note that the methods developed in all three parts could be used independently. Although the mapping framework developed in part one is particularly well-suited as the backbone for parts two and three, both planners are general and could also be used with other hierarchical volumetric mapping

frameworks, such as Octomap [4]. In the following, we summarize the ideas behind each part.

1.3.1 Volumetric mapping

While working toward the first research objective, we identified the Multi-Resolution Analysis (MRA) conditions, formalized by Mallat and Meyer [5], as a rigorous mathematical definition for the desired behavior of multi-resolution volumetric maps. One way to guarantee that the MRA conditions are always satisfied is to represent the volumetric map using a wavelet basis. As motivated in Section 1.1, existing volumetric mapping approaches are constrained by memory, computational, and accuracy bottlenecks. Wavelet decompositions make it possible to represent volumetric maps with state-of-the-art compression rates without introducing noticeable query overheads or impacting accuracy, as they are lossless. We further leverage the fact that they guarantee that all resolution levels implicitly remain synchronized at all times to derive a coarse-to-fine measurement integration scheme. By progressively increasing the update resolution only where it is needed, this integrator simultaneously increases the computational efficiency and achievable accuracy. Finally, we propose a novel measurement model that explicitly models the angular uncertainty of each measurement ray. From a theoretical perspective, removing the assumption that rays are infinitely thin allows us to directly adapt the resolution of map updates based on their approximation error. In practice, this new model significantly increases occupancy recall on thin objects, benefiting the safety of downstream tasks.

1.3.2 Local planning

A question that arises naturally is how multi-resolution can be leveraged to efficiently and accurately summarize subsections of a volumetric map for downstream tasks. One particularly challenging and relevant problem is summarizing a robot's direct surroundings for collision avoidance. Ideally, it would be possible to update such a summary with low latency, at sensor rate, and efficiently enough to allow the entire obstacle avoidance process to continuously run in the background. Intuitively fine details are relevant when they are near the robot, but lower resolutions suffice for distant objects. This is convenient, as it makes it possible to simultaneously consider thin local obstacles while also keeping a wide perceptive radius. Based on this insight, we propose a hierarchical algorithm that efficiently summarizes the obstacles in the robot's vicinity as multi-resolution cubes whose size decays in function of their distance to the robot. Leveraging the concept of Riemannian Motion Policies (RMPs),

we then attach one policy to each multi-resolution obstacle and combine them in parallel to obtain a highly efficient reactive collision avoidance planner. Through numerical derivations, we analyze the approximation error introduced by considering distant obstacles at a lower resolution. We then demonstrate how our reactive collision avoidance policy can easily be combined with additional RMPs to satisfy additional objectives, such as goal-seeking, to form a complete navigation system. Extensive evaluations are performed in simulation to analyze how the system compares to existing methods in terms of latency, success rates, and resource usage. Finally, the system is used to guide a real Micro Aerial Vehicle (MAV) through an obstacle course to study how it performs in practice.

1.3.3 Global planning

In the final research segment of this thesis, we investigate how multi-resolution can be used to increase the efficiency and efficacy of global path planning. Our focus centers on search-based methods, due to their deterministic nature and their ability to either find the optimal solution or report that the planning query is infeasible in finite time. A major drawback of using search-based planners is their execution time, in contrast to sampling-based planners which are significantly faster in large, open environments. Drawing inspiration from the ability of hierarchical volumetric maps to completely yet compactly capture an environment's inherent structure, we study whether multi-resolution can also be used to increase the efficiency of search-based planning. Concretely, we introduce a multi-resolution extension of Theta* [6], a slow but accurate any-angle planner. Using a dynamic, resolution refinement scheme and a special initialization procedure, our multi-resolution planner efficiently searches the shortest path in a coarse to fine manner, while directly controlling its worst-case sub-optimality with respect to Theta* running at the highest resolution. We extensively evaluate our proposed multi-resolution planner on a variety of real maps – from tight indoor spaces to large, structured, and unstructured outdoor environments. The critical role of each component of our approach is quantified through ablations. Finally, we compare the success rates, path quality, and execution times of our multi-resolution planner to a representative range of search and sampling-based planners to get a comprehensive understanding of its performance.

1.4 Contributions

The core contributions of this thesis are organized into three parts, corresponding to the research chapters of this thesis:

- **Volumetric mapping**
  The main contribution of this thesis is *wavemap*, a hierarchical volumetric mapping framework inspired by multi-resolution analysis. The MRA theory guarantees that when the map is represented using wavelet decomposition, new measurements can safely and very efficiently be integrated in a coarse-to-fine manner. The resulting gains in computational efficiency, together with early stopping criteria for the integrator, allow us to use more complex sensor models. We therefore propose to use a new angular uncertainty-aware model to increase accuracy. In experiments on synthetic RGB-D and real-world 3D LiDAR data, we demonstrate that our proposed method achieves high-quality results while being efficient in terms of memory and compute requirements. We also demonstrate how our method can incorporate observations from multiple sensors into a single map with per-sensor resolution. This allows the use of a single map representation for tasks that would have required several dedicated maps in the past. This work is the subject of publication [7].

- **Local planning**
  Our second contribution is a method to enable efficient, reactive obstacle avoidance in changing or unknown 3D environments. Following the intuition that distant geometry does not need to be considered at the same resolution as nearby obstacles, it leverages multi-resolution to summarize the robot's surroundings. A hierarchical algorithm is presented to efficiently extract multi-resolution obstacle summaries from volumetric mapping frameworks such as *wavemap*. We then show how the summary can be turned into a reactive collision avoidance policy using RMPs. Numerical derivations show that reducing the resolution of distant obstacles makes it possible to consider a much larger perceptive radius, while only introducing a negligible approximation error. An important advantage of using RMPs is that it makes it easy to combine our reactive collision avoidance policy with RMPs satisfying additional objectives, such as goal-seeking, to form a complete navigation system. Extensive statistical evaluations on indoor and outdoor maps show that the proposed system performs on par with optimization-based planners such as CHOMP [8] while reducing the planning time by $50\times$ and requiring no pre-processing or post-processing steps, such as Euclidean Signed Distance Field (ESDF) generation and trajectory tracking control. Finally, we deploy the system on a real MAV with an Nvidia

Jetson AGX Orin and show that it runs at 200Hz, with an end-to-end latency of 36ms, using only 2.4 threads for mapping and planning. The drone is shown to successfully negotiate an indoor obstacle course. This work is the subject of publication [9].

- **Global planning**
  The final contribution of this thesis is an efficient, accurate, and complete global planner. We start by studying how multi-resolution can be used to store the search algorithm's intermediate solutions more efficiently and introduce a representation that combines the efficiency of octrees with the accuracy of any-angle planning representations such as Theta* [6]. We then introduce a complementary algorithm that efficiently explores the search space by exploiting the inherent structure of hierarchical occupancy maps. The algorithm operates in a coarse to fine manner and dynamically increases the resolution where needed to control its worst-case sub-optimality. A special initialization procedure is introduced to close the accuracy gap between our multi-resolution planner and Theta*, without compromising its efficiency. Extensive evaluations are performed across a variety of real indoor and outdoor environments. Through ablations, we quantify the importance of our method's core components and show how they allow users to intuitively trade off some optimality for lower runtimes. We further compare our proposed planner to a range of well-established search and sampling-based planners. The results show that our method reliably finds shorter paths than RRTConnect and RRT* in all environments while achieving significantly higher success rates in confined spaces. We also show that our multi-resolution planner empirically maintains the completeness guarantees of search-based planners running at the highest resolution, while being significantly faster. In particular, our method is up to 3 orders of magnitude faster than Theta* when allowed to find solutions that are longer by 2% at worst and 0.1% on average.

1.4.1 Publications

This section lists the publications to which the author contributed during his doctorate.

Discussed in this Thesis

The following publications are directly related to the main contributions of this thesis and their results are extended in the document at hand.

- V. Reijgwart, C. Cadena, R. Siegwart and L. Ott, "Efficient volumetric mapping of multi-scale environments using wavelet-based compression," RSS 2023
- V. Reijgwart*, M. Pantic*, R. Siegwart, L. Ott, "Waverider: Leveraging Hierarchical, Multi-Resolution Maps for Efficient and Reactive Obstacle Avoidance," ICRA 2024

Related Publications

The results of the following publications motivate the work presented in this thesis. However, none of their content is included here.

- V. Reijgwart*, A. Millane*, H. Oleynikova, R. Siegwart, C. Cadena and J. Nieto, "Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps," in RA-L 2020
- L. Gasser, A. Millane, V. Reijgwart, R. Bähnemann and R. Siegwart, "Voxplan: A 3D Global Planner using Signed Distance Function Submaps," ICRA 2021
- L. Schmid*, V. Reijgwart*, L. Ott, J. Nieto, R. Siegwart and C. Cadena, "A Unified Approach for Autonomous Volumetric Exploration of Large Scale Environments Under Severe Odometry Drift," RA-L 2021
- M. Kulkarni et al., "Autonomous Teamed Exploration of Subterranean Environments using Legged and Aerial Robots," ICRA 2022
- M. Tranzatto et al., "CERBERUS: Autonomous Legged and Aerial Robotic Exploration in the Tunnel and Urban Circuits of the DARPA Subterranean Challenge," Field Robotics 2022
- M. Tranzatto et al., "Team CERBERUS Wins the DARPA Subterranean Challenge: Technical Overview and Lessons Learned," In review

Other Publications

The author also contributed to the following publications which are not directly related to the work presented in this thesis.

- J. Kabzan et al., "AMZ Driverless: The full autonomous racing system," Journal of Field Robotics 2020
- L. Andresen et al., "Accurate Mapping and Planning for Autonomous Racing," IROS 2020

- S. Srinivasan, I. Sa, A. Zyner, V. Reijgwart, M. Valls and R. Siegwart, "End-to-End Velocity Estimation for Autonomous Racing," RA-L 2020
- P. Pfreundschuh, H. Hendrikx, V. Reijgwart, R. Dubé, R. Siegwart and A. Cramariuc, "Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data," ICRA 2021
- A. Cramariuc et al., "maplab 2.0 – A Modular and Multi-Modal Mapping Framework," RA-L 2023

1.4.2 Open-Source Software

Reference implementations for all of the works discussed within this thesis have been released open-source, to allow others to build on its results. The author also released or contributed to the development of several other open-source packages.

- Voxblox [10]: a fixed-resolution, voxel-based volumetric mapping library focusing on truncated and Euclidean distance fields.
  https://github.com/ethz-asl/voxblox
- A package to generate ground truth volumetric maps from meshes or simulated environments, used for accuracy evaluations [3] and simulations [11].
  https://github.com/ethz-asl/voxblox_ground_truth
- A package to motion-undistort LiDAR pointclouds based on an odometry input.
  https://github.com/ethz-asl/lidar_undistortion
- Voxgraph [3]: a globally consistent volumetric mapping framework leveraging a collection of submaps aligned through graph optimization.
  https://github.com/ethz-asl/voxgraph
- GLocal [1]: a framework for efficient volumetric exploration planning under severe odometry drift.
  https://github.com/ethz-asl/glocal_exploration
- COHORT: a multi-agent volumetric exploration planner, presented together with GBPlanner 2.0 [2].
  https://github.com/ntnu-arl/cohort_exploration
- Wavemap [7]: an efficient and accurate multi-resolution, multi-sensor 3D occupancy mapping framework, presented in Chapter 3.
  https://github.com/ethz-asl/wavemap

- Waverider [9]: an efficient, low-latency reactive collision avoidance planner with a wide perceptive radius, presented in Chapter 4.
  https://github.com/ethz-asl/waverider
- Wavefinder: an efficient and complete search-based global planner leveraging multi-resolution, presented in Chapter 5.
  https://github.com/ethz-asl/wavefinder

1.4.3 Teaching and Student Supervision

During his doctoral studies, the author had the opportunity to be a teaching assistant in "Autonomous Mobile Robots" (2020, 2021, 2022) and "Robot Dynamics" (2019, 2023). Furthermore, he supervised the following student projects:

Master Theses, 6-month, full-time

- Bagheri, Davide (Spring 2020): "Autonomous globally consistent exploration using submaps collections"
- Gasser, Laura (Fall 2019): "Global Path Planning in SDF Submaps"
- Camus, Amaury (Fall 2019): "Optimization-Based Torso Trajectory Planning for Online Obstacle Avoidance"
- Dall'Olio, Alberto (Fall 2019): "Adaptive Deep Stereo Network for Collision Avoidance"
- Pfreundschuh, Patrick (Spring 2020): "Dynamic Object Detection for Robust and Accurate LiDAR SLAM"
- Brits, Sonja (Spring 2020): "Semantics-Based Localization"
- Gulich, Lionel (Spring 2021): "Navigation Planning for wheeled Robots in multi-layered Environments"
- Brandemuehl, Adrian (Spring 2021): "Tightly Coupled LiDAR-Visual-Inertial Odometry"
- Phillips, Trevor (Spring 2021): "Vision-based 3D Perception for Aerial Terrain Reconstruction"
- Pasini, Gianni (Spring 2021): "Autonomous Exploration in Confined Spaces with a Tiltable Tri-Copter"

- Cheema, Mansoor (Fall 2021): “Leveraging Deep Learnt Scene Completion for Fast Exploration Planning and Mapping”
- Marti, Dominic (Fall 2021): “Underwater Volumetric Occupancy Mapping with Imaging Sonar”
- Anthanasiadis, Ioannis (Spring 2022): “Towards Spatio-temporally Consistent Volumetric Mapping using Panoptic Submaps with Plane Constraints”

Perception and Learning for Robotics (PLR) student project, 6-months, 4 ECTS credits

- Kieffer Max and Marc Zünd (Spring 2024): “Neural Environment Encoding with Hierarchical Map Representations”

1.5 Organization

This thesis is organized into six chapters. Chapter 2 provides a brief introduction to the theory behind Multi-Resolution Analysis and orthogonal wavelet bases. Building on these mathematical tools, the main contribution of this thesis, a hierarchical volumetric framework, is presented in Chapter 3. We then investigate how hierarchical volumetric maps can be used to increase the efficiency of two representative downstream tasks: local planning, in Chapter 4, and global planning, in Chapter 5. Finally, Chapter 6 summarizes the main findings of this thesis and provides an outlook for possible future research directions.

Chapter 2

From Multi-Resolution Analysis to Wavelets

This chapter is intended as a primer on wavelet theory, providing additional context for Chapter 3. We start with a short introduction to the MRA conditions, before showing how orthogonal wavelet bases fulfill these requirements. We then discuss how wavelet decompositions can efficiently be computed using the Fast Wavelet Transform. For readers who are interested in learning more about sparse signal processing using wavelets, we warmly recommend [5, 12]. Note that an intuitive explanation of the MRA conditions in the context of volumetric mapping is provided in Section 3.3.

2.1 Multi-Resolution Analysis

Multi-resolution representations are regularly used in the context of computer vision and robotics. For example, in Laplacian image pyramids introduced by Burt and Adelson [13]. Mallat and Meyer [5], formalized the expected behavior of multi-resolution representations through the MRA conditions:

$$\forall (j,k) \in \mathbb{Z}^2, \qquad f(x) \in V_j \Leftrightarrow f(x - 2^j k) \in V_j \tag{2.1}$$

$$\forall j \in \mathbb{Z}, \qquad V_{j+1} \subset V_j \tag{2.2}$$

$$\forall j \in \mathbb{Z}, \qquad f(x) \in V_j \Leftrightarrow f(x/2) \in V_{j+1} \tag{2.3}$$

$$\lim_{j \to \infty} V_j = \bigcap_{j=-\infty}^{\infty} V_j = \{0\} \tag{2.4}$$

$$\lim_{j \to -\infty} V_j = \text{closure}\left(\bigcup_{j=-\infty}^{\infty} V_j\right) = L^2(\mathbb{R}) \tag{2.5}$$

$$V_0 \text{ admits a Riesz basis} \tag{2.6}$$

where the sequence of subspaces $\{V_j\}_{j \in \mathbb{Z}}$ corresponds to the map's representations at increasing resolution levels 2^j, referred to as scales, and each V_j is a closed subspace of Lebesgue space L^2. Starting with condition 2.6, the most common Riesz basis used in robotics consists of box functions arranged to span the cells of a regular grid. In this case, the scale 2^j corresponds to the cell width. Condition 2.1 ensures self-similarity in space. Specifically, if subspace V_j can represent function $f(x)$, it can also represent the same function shifted by integer multiple of the cell size. Condition 2.2 states that the subspaces are nested. In other words, any function contained in subspace V_{j+1} must also be contained in next finer subspace V_j and by extension in all finer subspaces. Condition 2.3 ensures self-similarity in scale. If V_j contains $f(x)$, V_{j+1} must be able to contain $f(x)$ dilated by 2. Finally, conditions 2.4 and 2.5 ensure completeness. At the coarsest scale ($j \to \infty$), V_j only contains the zero element, whereas refining the scale ($j \to -\infty$) eventually allows us to represent any signal in L^2.

2.2 Orthogonal wavelet bases

The principal idea behind wavelets is that they represent the difference between the consecutive resolutions of a signal's MRA. Formally, they span a second subspace W_j which is the orthogonal complement to V_j, such that $V_j \oplus W_j = V_{j-1}$ where $\oplus$ is the vector-space direct sum operator. In words, this means that by combining a signal's representation V_j with its wavelet details at the same resolution, W_j, we obtain the signal's representation at the next higher resolution V_{j-1}.

An orthogonal basis for all V_j can be obtained by translating and dilating a single function ϕ, referred to as the scaling function, as $\phi_{jk}(x) = \frac{1}{2^j}\phi(\frac{x-2^j k}{2^j})$. The scaling function can be found by orthogonalizing the Riesz basis of V_0 as described in [5]. In similar fashion, an orthogonal basis for W_j can be obtained by translating and scaling a single wavelet function ψ as $\psi_{jk}(x) = \frac{1}{2^j}\psi(\frac{x-2^j k}{2^j})$. One condition that any wavelet function has to fulfill in order to be admissible is that its average must be zero $\int_{-\infty}^{\infty} \psi(x)dx = 0$. More generally, the scaling functions and wavelet functions can be seen as complementary low and high-pass filters that, when combined, can perfectly reconstruct the signal from the next finer scale. Since wavelet bases form a valid MRA, this concept can be applied recursively and the entire map can be represented

by stacking a single scaling function at the coarsest scale with a hierarchy of wavelet functions at increasing scales.

Note that the Riesz basis consisting of box filters arranged to span the cells of a regular grid, mentioned previously, is already orthogonal. In fact, the unit box filter can be used as a scaling function

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

and doing so directly leads to the Haar basis [5]. The corresponding Haar wavelet function can be derived by finding ϕ's orthogonal complement while enforcing the MRA conditions and is given by

$$\psi(x) = \begin{cases} -1 & 0 \leq x < 1/2 \\ 1 & 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

Orthogonal wavelet bases of $\mathbb{R}$ can be extended to separable orthogonal bases b for $\mathbb{R}^n$ by combining the scaling and wavelet functions along each dimension as

$$b = \left\{ \prod_{k=1}^{n} \phi(x_k)^{o_k} \psi(x_k)^{1-o_k} \right\}_{\forall o \in \{0,1\}^n} \tag{2.9}$$

2.3 The Fast Wavelet Transform

The discrete wavelet transform for a function f and wavelet ψ is defined as the projection of f onto the set of all integer scalings and translations of the wavelet function $\{\psi_{jk}\}_{j,k\in\mathbb{Z}}$. Each wavelet coefficient d_{jk} is thus computed as

$$d_{jk} = \sum_{x=-\infty}^{\infty} f(x)\frac{1}{2^j}\psi\left(\frac{x-2^j k}{2^j}\right) \tag{2.10}$$

where the summation could be replaced by an integral if the domain of f is real-valued instead of discrete. Note that this transform is linear and, for orthogonal wavelets, orthogonal.

The coefficients d_{jk} can efficiently be computed using the Fast Wavelet Transform (FWT) algorithm [5], which exploits the hierarchical MRA structure to remove redundant operations. The FWT is initialized by projecting f onto the scaling functions at the finest scale $a_{0k} = \sum_{-\infty}^{\infty} f(x)\phi\left(x-k\right)$ or with a good approximation thereof. At each iteration, these coefficients are then filtered and downsampled to obtain the wavelet and scaling coefficients at the next coarser scale. These iterations are typically repeated until only 1 scaling coefficient is left, or a desired number of levels is reached. For wavelets with finite spatial support and functions f sampled at N points, the FWT computes the full wavelet decomposition in $O(N)$ time. Extending the FWT to only (de)compress regions-of-interest or single cells is straightforward and very efficient if the spatial support of the chosen wavelet is small, as is the case for the Haar wavelet.

Chapter 3

Hierarchical volumetric mapping

Volumetric maps are widely used in robotics due to their desirable properties in applications such as path planning, exploration, and manipulation. Constant advances in mapping technologies are needed to keep up with the improvements in sensor technology, generating increasingly vast amounts of precise measurements. Handling this data in a computationally and memory-efficient manner is paramount to representing the environment at the desired scales and resolutions. In this chapter, we express the desirable properties of a volumetric mapping framework through the lens of multi-resolution analysis. This shows that wavelets are a natural foundation for hierarchical and multi-resolution volumetric mapping. Based on this insight we design an efficient mapping system that uses wavelet decomposition. The efficiency of the system enables the use of uncertainty-aware sensor models, improving the quality of the maps. Experiments on both synthetic and real-world data provide mapping accuracy and runtime performance comparisons with state-of-the-art methods on both RGB-D and 3D LiDAR data. The framework is open-sourced[1] to allow the robotics community at large to explore this approach.

The work described in this chapter is presented in the following publication:

- Reijgwart, V., Cadena, C., Siegwart, R., & Ott, L. (2023). Efficient volumetric mapping of multi-scale environments using wavelet-based compression. Proceedings of Robotics: Science and Systems

[1] https://github.com/ethz-asl/wavemap

3.1 Introduction

As robots move from tightly controlled spaces into our everyday lives, there is a growing need for them to autonomously navigate and work in increasingly large, unstructured, and unknown environments. For reliable deployments and robust operation over extended periods of time, robots need to build and maintain their representation of the world using only onboard sensing and computing. Doing this in a timely manner on compute restricted devices using sensors producing large amounts of data is a continual challenge in robotics.

Dense geometric environment representations are widely used to facilitate tasks ranging from navigation to inspection and manipulation, while also serving as building blocks for other representations. Robotics is a particularly challenging field for such representations, due to the demands placed on systems with limited computational resources. For example, building a map of an unknown environment while localizing in it with Simultaneous Localization And Mapping (SLAM) requires the ability to update the map incrementally at interactive rates. To support high-level tasks such as exploration and navigation the representation must also differentiate between unknown space and observed (free or occupied) space. Finally, the map must be able to model arbitrary geometry with sufficient accuracy to guarantee safety when unexpected environmental structures or objects are encountered.

Volumetric map representations can be updated incrementally and explicitly represent unknown space. Furthermore, if a sufficiently high resolution is chosen, they can also represent object surfaces and unknown space boundaries of arbitrary topology. Beyond robotics, volumetric representations are commonly used in 3D reconstruction, reality capture, and augmented reality applications. However, a major drawback of volumetric representations is that their memory usage in naive implementations grows linearly with the observed volume and cubically with the chosen resolution. Several research efforts propose to use multi-resolution representations, often based on trees, and demonstrate significant improvements. In this chapter, we extend these efforts by approaching the problem from a formal signal processing and data compression perspective. Specifically, we propose to use wavelet compression to obtain a hierarchical volumetric representation. Using Haar wavelets we achieve state of the art lossless compression performance, while also allowing simple yet efficient updates and queries. This is achieved by compressing the occupancy information using a Haar wavelet decomposition and storing the individual decomposition components in a hierarchical data structure. The wavelet transform's linearity makes it possible to perform measurement updates directly in the map's compressed representation. Furthermore, when performing map updates we know that all resolution levels of the map

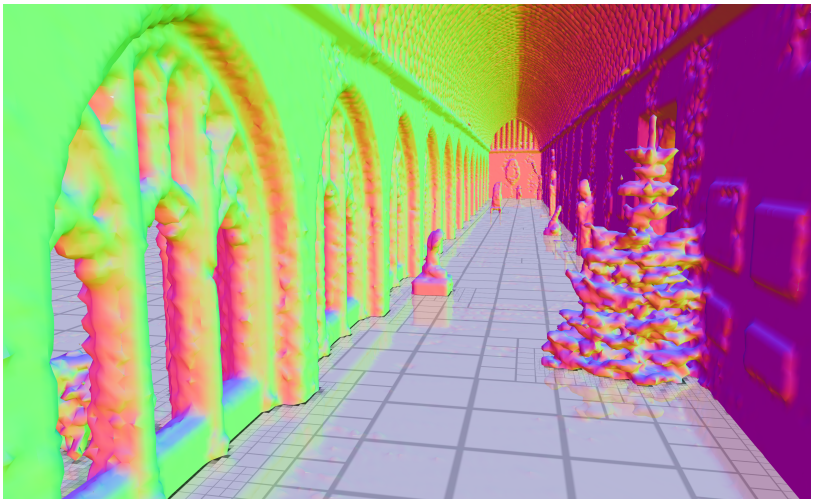


Figure 3.1: A reconstruction created by our proposed hierarchical volumetric mapping framework, *wavemap*, highlighting its ability to accurately capture fine objects while also efficiently compressing free space as shown by the adaptive resolution along the transparent slice.

are always up to date and in a valid state due to the Haar basis' orthogonality property. This obviates the need to perform maintenance operations or manual compression passes that are typically seen in other multi-resolution mapping frameworks.

Another trade-off made by many existing volumetric mapping methods is the reliance on simplified measurement models to achieve real-time update rates. A common approach is to use discrete occupancy updates, that systematically inflate obstacles and do not allow for surfaces to be reconstructed with sub-voxel accuracy. Measurement models based on Truncated Signed Distance Fields (TSDFs) overcome the latter limitation but use a projective distance heuristic. Such approaches have a hard time reconstructing thin objects such as branches, cables, or fences. Furthermore, the implied assumption of infinitely thin rays, underlying these observation models, leads to aliasing artifacts in regions where the ray density is low compared to the voxel resolution. In addition to negatively affecting the reconstruction quality, the resulting high entropy regions are hard to compress. Besides alleviating the challenges

mentioned above, modeling soft beams provides an opportunity to incorporate angular uncertainties from sensor calibration and pose estimation into the volumetric reconstruction process. Thanks to the computational benefits of the Haar wavelet representation we can adopt a continuous occupancy measurement model, accounting for angular and range uncertainty, inspired by the work of [14].

In order to process data at sensor rate we introduce a specialized measurement integration algorithm that exploits a hierarchical measurement update approach with the information provided by the map itself. The proposed algorithm speeds up measurement integration while guaranteeing that the results are identical to a naive integrator applying the same measurement updates at the highest resolution throughout the field of view.

In summary, the main contribution of this chapter is a volumetric mapping system that uses:

- A wavelet-based hierarchical representation, that is guaranteed to keep the hierarchy consistent at all times;
- A continuous occupancy measurement model accounting for range and angular uncertainties;
- A highly-efficient coarse-to-fine measurement integrator that adapts to the observed structure;

The proposed framework is extensively evaluated on synthetic and real-world datasets with comparisons to several state-of-the-art methods. The results demonstrate that our approach is memory efficient yet produces high-quality maps, all while being computationally efficient. The entire framework is open-sourced to enable the robotics community to build on these results.

3.2 Related work

3.2.1 Map model

Two approaches are commonly used to represent maps in robotics [15], sparse feature-based maps and dense maps. The first category uses sparse sets of distinctive features [16, 17] and excels at representing large environments but struggles to model the connectivity of surfaces and distinguish between free and unknown space. This makes it ideal for large scale mapping and localization tasks, but limits its use for manipulation, motion planning, and exploration tasks. The second paradigm uses a

large number of geometric elements, such as as points [18, 19], surfels [20–24], or meshes [25] to model observed obstacles. Voxels, discretizing the space into squares or cubes of fixed size, are another common geometric primitive used to model both occupancy [4, 26] and signed distance information [27–30].

3.2.2 Measurement model

Approximations of the sensor's physical operation have been widely explored. Early approaches modeled uncertainties of the sensors explicitly [26]. Other approaches aim to achieve specific map properties, such as sharp map boundaries [14]. However, when building 3D maps using precise sensors the computational cost incurred by these sensor models motivated the development of simpler ray-based models. These models treat observations as thin rays tracing through the world [4, 29]. Machine-learning based methods exploit more complex relationships, such as inverse rendering [31, 32] or beam-to-beam interactions [33].

3.2.3 Map storage

The most common way to store volumetric maps is to discretize the space using a voxelgrid, i.e. a regular grid with fixed size voxels. In the beginning grids with a single fixed resolution [26, 28] were used, but over time spatial data structures, such as hashed voxel blocks [34], trees [4], or hybrids thereof [35] were adopted. These structures fit the observed volume more tightly, can grow dynamically, and improve runtime. To model expansive maps with varying levels of detail, multi-resolution maps [4, 36–39] are widely used due to being memory efficient and capable of adapting to the needed resolution. Many multi-resolution representations are also hierarchical, allowing users to query the map at varying resolutions [4, 38]. Taking a signal processing perspective on compact map storage leads to the use of wavelet transforms [40], which are inherently multi-resolution and hierarchical, or the discrete cosine transform [41]. Recent learning-based methods, such as NERF [31] or occupancy prediction networks [42, 43], take a different approach and learn the coefficients of a neural network that predicts map information at arbitrary coordinates.

3.2.4 Map updates

The manner in which maps are updated with new observations is crucial for the efficiency and map quality. Early volumetric frameworks evaluated the measurement model for all voxels in the observed volume [26, 28]. This was improved by tracing

rays from the sensor's center to each measured point and updating the voxels that are intersected by the ray [4, 29]. Advances in sensor technology, enabling high resolution maps spurred further efficiency improvements, such as ray-tracers that bundle [29] or sub-sample [39, 44] similar rays, or rate limit voxel updates [44]. While efficient, these integrators can produce "holes" in the map depending on the sensor's ray density. This motivates the use of projective integrators which avoid this problem by interpolating the depth image [14, 27]. Other approaches to avoid resolution-related issues include multi-resolution integrators, ray-tracing [39] or projective [37], which reduce the update resolution with distance, as well as methods analyzing the measurement update regularity [38, 45]. While efficient, hierarchical volumetric maps require maintenance to keep the information in the different levels coherent. Octomap [4] employs a fine-to-coarse scheme, integrating measurements at the finest resolution and synchronizing coarser levels in a maintenance pass. Supereight [37, 38] performs multi-resolution updates and synchronizes the remaining levels using an upward and downward propagation scheme.

In contrast to others, our volumetric ray-tracing method uses a wavelet decomposition-based representation which implicitly synchronizes all hierarchy levels at once. Additionally, unlike most ray-tracing methods we use a continuous sensor model, taking angular uncertainty into account, to improve map accuracy.

3.3 Multi-Resolution Analysis and Wavelets

Multi-resolution representations have been the subject of intensive study by communities ranging from computer vision [13] to physics and mathematics [5, 12]. Mallat and Meyer formalized the expected properties of multi-resolution representations as the MRA conditions [5]. The full MRA conditions are summarized in Section 2.1. In informal terms, they state that increasing the resolution should only add detail and eventually make it possible to represent any signal. Two further requirements are self-similarity in space and in scale. In a mapping context, these imply that the map should behave the same regardless of our frame of reference and choice of units.

A corollary of the fact that increasing the resolution only adds information is that, in areas that are stored at multiple resolutions, the lower resolutions do not carry any unique information and storing them explicitly is redundant. This motivates the use of wavelet decompositions, which allow us to work with maps that form valid MRAs while only storing and processing the differences between the resolution levels. A given wavelet decomposition is characterized by its chosen scaling function and complementary wavelet function. In this work, we focus on the Haar wavelet

and scaling function, which form an orthogonal basis. A summary of orthogonal wavelet bases is provided in Section 2.2. This orthogonality is particularly beneficial because it guarantees that any given volumetric map is characterized by a unique combination of wavelet coefficients. Thus, there are no redundant coefficients that can go out of sync and manually have to be updated after integrating new measurements. Another interesting property of Haar wavelets is that the basis resulting from its scaling functions correspond to box functions arranged to span the cells of a regular grid. Therefore, Haar decompositions can represent anything a regular grid map can, while bringing significant benefits in terms of compression and implicitly maintaining the hierarchy's consistency.

3.4 Method

In the following, we describe the components of our approach. We first explain how the map's occupancy posterior can be efficiently updated in its compressed state, thanks to the properties of the wavelet transform. Next, we derive our continuous sensor model, which captures range and angular uncertainties associated with the measurements. After that, we derive an error bound which enables early stopping during the coarse-to-fine observation integration process. Further performance improvements are obtained by skipping updates that do not change the state of the map. Finally, we illustrate how all these pieces fit together with an algorithmic overview.

3.4.1 Measurement integration

In the following we will explain how the use of wavelets enables efficient measurement integration. As each new beam endpoint measurement $\mathbf{z}$ arrives, the map's Bayesian occupancy posterior $p(m_{\mathbf{x}}|\mathbf{z}_{1:t})$, estimated at each point $\mathbf{x}$ in the map m, can incrementally be updated using

$$\mathcal{L}_p(m_{\mathbf{x}}|\mathbf{z}_{1:t}) = \mathcal{L}_p(m_{\mathbf{x}}|\mathbf{z}_{1:t-1}) + \mathcal{L}_s(m_{\mathbf{x}}|\mathbf{z}_t), \tag{3.1}$$

where $s(m_{\mathbf{x}}|\mathbf{z}_t)$ is the sensor's inverse measurement model and the log-odds formulation, $\mathcal{L}_p = \log \frac{p}{1-p}$, is used to make the update linear. As the wavelet transform $\mathcal{W}$ is also linear, the update equation for all cells in the map becomes:

$$\mathcal{W}\left(\mathcal{L}_p(m|\mathbf{z}_{1:t})\right) = \mathcal{W}\left(\mathcal{L}_p(m|\mathbf{z}_{1:t-1})\right) + \mathcal{W}\left(\mathcal{L}_s(m|\mathbf{z}_t)\right). \tag{3.2}$$

Therefore, once the compressed measurement update $\mathcal{W}\left(\mathcal{L}_s(m|\mathbf{z}_t)\right)$ is computed, the map can be updated directly in wavelet space. This is avoids the costly process,

employed by other methods, of decompressing the map's observed area, applying the update, and compressing the map again. Computing $\mathcal{W}\left(\mathcal{L}_s(m|\mathbf{z}_t)\right)$ is efficient thanks to the FWT (Section 2.3), which is typically initialized by computing the orthogonal projection of the original signal onto the scaling functions at a pre-determined finest resolution.

Since the wavelet transform itself is lossless, the reconstruction error is fully determined by how well the initial FWT projection approximates the original update. Most applications use a constant finest resolution, but this is not mandatory. Given that inverse sensor models tend to be smooth throughout most of the observed volume, only raising the resolution close to surfaces would improve efficiency and the maximum achievable detail.

3.4.2 Measurement models

In order to derive multi-resolution sampling and integration approaches, it is important that the chosen inverse measurement model $s(m_{\mathbf{x}}|\mathbf{z}_t)$ is well-defined at all points $\mathbf{x}$ in the observed volume. We propose to extend the continuous occupancy model introduced in [14] by modeling the angular uncertainty of each measured beam, in addition to range uncertainty. We model the probability of occupancy $s(m_{\mathbf{x}}|\mathbf{z})$ at a point $\mathbf{x}$ for a single beam $\mathbf{z}$ by correlating the probability of occupancy given the beam's true endpoint $\bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}})$ with the distribution of the true endpoint position given a noisy observation $o(\bar{\mathbf{z}}|\mathbf{z})$, i.e.:

$$s(m_{\mathbf{x}}|\mathbf{z}) = \int_{\mathbb{S}} \bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}}) o(\bar{\mathbf{z}}|\mathbf{z}) d\bar{\mathbf{z}}, \tag{3.3}$$

where $\mathbf{x}$, $\bar{\mathbf{z}}$, and $\mathbf{z}$ are expressed in sensor coordinate space $\mathbb{S}$, and the beam start point is at its origin. Extending [14] to include angular uncertainty, we define $\bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}})$ as

$$\begin{aligned} \bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}}) &= \bar{s}(m_{\mathbf{x}}|\bar{z}_r, \bar{z}_\theta) \\ &= \begin{cases} 0 & x_r < \bar{z}_r \ \wedge \ |x_\theta - \bar{z}_\theta| \leq \tau_\theta \\ 1 & \bar{z}_r \leq x_r \leq \bar{z}_r + \tau_r \ \wedge \ |x_\theta - \bar{z}_\theta| \leq \tau_\theta \\ \frac{1}{2} & \text{otherwise} \end{cases} \end{aligned} \tag{3.4}$$

where τ is an assumed surface thickness parameter in sensor coordinates, see Fig. 3.2a for a visualization. The subscript r refers to the axis perpendicular to the sensor's image plane, whereas θ refers to the offset along the image plane[2].

[2] For pinhole camera projection models, r corresponds to the depth coordinate and θ to the reprojection error. For spherical projection models, e.g. certain LiDARs, r refers to the range coordinate and θ to the relative angle.

Our model assumes that the noise on the measurement beam endpoint position is normally distributed in sensor coordinates, as

$$o(\mathbf{z}|\bar{\mathbf{z}}) \sim \mathcal{N}(\bar{\mathbf{z}}, \Sigma), \tag{3.5}$$

where Σ is the measurement noise covariance matrix. If Σ is diagonal and $\bar{\mathbf{z}}$ has a uniform prior, the r and θ components are independent and eq. 3.5 can be simplified as follows:

$$o(\bar{\mathbf{z}}|\mathbf{z}) = o(z_r|\bar{z}_r)o(z_\theta|\bar{z}_\theta) = \mathcal{N}(\bar{z}_r, \sigma_r)\mathcal{N}(\bar{z}_\theta, \sigma_\theta). \tag{3.6}$$

We approximate the normal distributions with quadratic B-splines, as in [14], such that $o(z|\bar{z}) \simeq q(\frac{\bar{z}-z}{\sigma})$, where

$$q(t) = \begin{cases} \frac{1}{16}(3+t)^2 & -3 \leq t \leq -1 \\ \frac{1}{8}(3-t^2) & -1 < t < 1 \\ \frac{1}{16}(3-t)^2 & 1 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases}. \tag{3.7}$$

The distribution of the true beam endpoint position given a noisy measurement (Fig. 3.2b) then becomes:

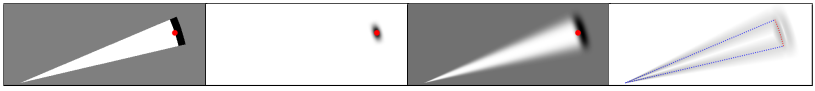
$$o(\bar{\mathbf{z}}|\mathbf{z}) = q\left(\frac{z_r - \bar{z}_r}{\sigma_r}\right) q\left(\frac{z_\theta - \bar{z}_\theta}{\sigma_\theta}\right). \tag{3.8}$$

As motivated in [14], we match the surface thicknesses to half the width of their respective B-splines, i.e. $\tau_r = 3\sigma_r$ and $\tau_\theta = 3\sigma_\theta$. This ensures that the measurement model is continuous and that $\mathcal{L}_p(m_\mathbf{x}|\mathbf{z}_{1:t})$ converges to 0 if $\mathbf{x}$ lies on an object's surface.

After substituting 3.4 and 3.8 into 3.3, the full inverse measurement model (Fig. 3.2c) becomes:

$$\begin{aligned} s(m_\mathbf{x}|\mathbf{z}) &= \int_0^\infty \int_{-\infty}^\infty \bar{s}(m_\mathbf{x}|\bar{z}_r, \bar{z}_\theta) q(v) q(w) d\bar{z}_\theta \, d\bar{z}_r \\ &= \frac{1}{2} + \left(Q(v) - \frac{Q(v-3)}{2} - \frac{1}{2}\right)\Big(Q(w+3) - Q(w-3)\Big), \end{aligned} \tag{3.9}$$

where $Q(t)$ refers to the cumulative distribution of $q(t)$, i.e. the cubic B-splines resulting from $Q(t) = \int_{-\infty}^t q(u)du$, and $v = \frac{z_r - \bar{z}_r}{\sigma_r}$, $w = \frac{z_\theta - \bar{z}_\theta}{\sigma_\theta}$.



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

Figure 3.2: Figure illustrating our proposed models for a) the occupancy given the true beam endpoint $\bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}})$ (eq. 3.4), b) the position of the true endpoint given a noisy measurement $o(\bar{\mathbf{z}}|\mathbf{z})$ (eq. 3.8), c) the complete inverse measurement model $s(m_{\mathbf{x}}|\mathbf{z})$ (eq. 3.9), and d) the local maxima used to derive the worst-case error bounds. Values of 0.0, 0.5 and 1.0 are shown in white, grey and black, respectively. The true beam endpoint is indicated in red. Uncertainties are exaggerated for illustration.

For depth cameras, the depth uncertainty is often set as $\sigma_r(x) = \kappa_r x_r^2$, where κ_r depends on the sensor setup and post-processing algorithms. For laser-based sensors, the range error is usually assumed to not vary with range, thus $\sigma_r = \kappa_r$ where κ_r is indicated on the sensor's datasheet.

Note that the shape of our proposed model resembles the original occupancy measurement model proposed by Elfes [26] for 2D sonars, which also considered angular uncertainty. The key difference is that Elfes' model reaches its peak, or maximum occupancy update, at the measured endpoint. As motivated by Loop et al. [14], this modeling decision inflates obstacles, resulting in biased occupancy maps. In contrast, our model extends [14], which peaks slightly behind the surface, and preserves its property that $s(m_{\mathbf{x}}|\mathbf{z})$ is exactly 0.5 when $x_r = z_r$, i.e. at the endpoint, such that $\mathcal{L}_p(m_{\mathbf{x}}|\mathbf{z}_{1:t})$ converges to 0 if $\mathbf{x}$ lies on an object's surface. In addition to reducing biases, this property simplifies surface reconstruction, as the surface then directly corresponds to the map's 0-level iso-surface, which is straightforward to extract.

3.4.3 Worst-case update error bounds

From the MRA theory (Section 3.3) we know that at some point, integrating information at finer levels of the hierarchy no longer improves the representation. Therefore, to fully exploit the coarse-to-fine measurement integration scheme of our method, we need to know at what level of the hierarchy we can stop integrating data. This requires determining, for each point $\mathbf{x}$, the resolution beyond which no further improvements are possible, which we achieve by deriving a conservative approximation error bound. As this work focuses on the use of Haar wavelets, we can exploit a property unique to them, namely that neighbors at the same resolution do not overlap. This results in the

leaves of our multi-resolution Haar decomposition perfectly partitioning the original space into non-overlapping cubes of varying sizes. Since Haar scaling functions are constant over their support, the worst-case error $\epsilon_{\max}$ within each space partition, or voxel, $\mathcal{V}$ is given by:

$$\epsilon_{\max}(\mathcal{L}_s(m, \mathbf{z}), \mathcal{V}) = \max_{\mathbf{x} \in \mathcal{V}} \left| \mathcal{L}_s(m_{\mathbf{x}'}, \mathbf{z}) - \mathcal{L}_s(m_{\mathbf{x}}, \mathbf{z}) \right|, \tag{3.10}$$

where $\mathbf{x}'$ is the chosen sample point, which we set to be the partition's center.

Since $\epsilon_{\max}$ has to be evaluated millions of times per second in practice, we simplify the computation by only considering three cases based on the state of the space partition, defined as follows:

$$\text{update\_type}(\mathcal{V}, \mathbf{z}_t) = \begin{cases} \text{FullyUnobserved} & \forall \mathbf{x} \in \mathcal{V} : \mathcal{L}_s(m_{\mathbf{x}} | \mathbf{z}_t) = 0 \\ \text{PossiblyOccupied} & \exists \mathbf{x} \in \mathcal{V} : \mathcal{L}_s(m_{\mathbf{x}} | \mathbf{z}_t) > 0 \\ \text{FreeOrUnobserved} & \text{otherwise} \end{cases} \tag{3.11}$$

Looking at Eq. 3.9 we can see that the gradient of $s(m_{\mathbf{x}}|\mathbf{z})$ is zero in FullyUnobserved areas and reaches local maxima where $x_\theta = z_\theta \pm 3\sigma_\theta$ or $x_r = z_r$ as illustrated in Fig. 3.2d. Using the fact that

$$\left.\frac{\partial s(m_{\mathbf{x}}|\mathbf{z})}{\partial x_\theta}\right|_{x_\theta = z_\theta \pm 3\sigma_\theta} = \frac{3}{16\sigma_\theta}, \quad \left.\frac{\partial s(m_{\mathbf{x}}|\mathbf{z})}{\partial x_r}\right|_{x_r = z_r} = \frac{3}{8\sigma_r} \tag{3.12}$$

and assuming the worst-case orientations for a cube-shaped partition $\mathcal{V}$, i.e. its diagonal projected into sensor coordinates r and θ aligns with either gradient, we obtain the following bounds for the approximation error for the three cases:

$$\epsilon_{\max}(\mathcal{V}) = \begin{cases} 0 & \text{FullyUnobserved} \\ \max\left(\frac{3\mathcal{V}_{h_\theta}}{16\sigma_\theta}, \frac{3\mathcal{V}_{h_r}}{8\sigma_r}\right) & \text{PossiblyOccupied} \\ \frac{3\mathcal{V}_{h_\theta}}{16\sigma_\theta} & \text{FreeOrUnobserved} \end{cases} \tag{3.13}$$

where $\mathcal{V}_h$ is the maximum distance a sample can have to $\mathcal{V}$'s center, namely half of $\mathcal{V}$'s diagonal. Note that $\mathcal{V}_{h_\theta}$ decays quickly as the distance to the sensor increases.

3.4.4 Saturated region skipping

To preserve the ability to quickly adapt the map when dynamic parts of the environment change, we impose upper and lower bounds on the occupancy posterior $\mathcal{L}_p(m_{\mathbf{x}}|\mathbf{z}_{1:t})$, as proposed by Yguel et al. [46]. As observed by Hornung et al. [4], this clamping policy also significantly improves compression performance by encouraging the majority of the map's posterior to converge to constant values. Namely to the lower bound in areas that are consistently observed as being free, and to the upper bound in areas that are consistently observed as being occupied. We propose to exploit this saturating behavior further to reduce the computational cost of map updates. Applying negative occupancy (free-space) updates in areas where the posterior has already reached the lower bound has no effect, as the updates are canceled out by the clamping operation. Similarly, the posterior is not affected by skipping positive occupancy updates in areas that already converged to the upper bound. Skipping saturated regions leads to a particularly high speedup if it can be done in a coarse-to-fine manner, but doing so is only safe if the map's lower resolutions are always up to date. Both properties are met by our representation and integration scheme. An algorithm that interleaves saturated region skipping, adaptive sampling, and thresholding will be discussed in the next section.

3.4.5 Algorithm and data structure

As described previously, Haar scaling functions do not overlap with their neighbors at the same resolution and perfectly partition the space. The support of the scaling functions in a multi-resolution Haar decomposition is, therefore, identical to the hierarchical partitioning scheme of octrees. We can thus store the wavelet coefficients in any optimized octree data structure that allows data to be attached to both inner and leaf nodes, such as supereight [36] or OpenVDB [47].

Leveraging the idea that increasing the resolution in MRAs only adds information, our proposed adaptive multi-resolution update algorithm determines the appropriate update resolution for all points in the observed volume in a coarse-to-fine manner (Section 3.4.3). The algorithm is initialized at the octree's root and recursively evaluates its children, as illustrated in Algorithm 1. Each recursive call starts by checking which of the three possible update cases, eq. 3.11, applies to the current node's partition $\mathcal{V}$. If no parts of the partition have been observed by the current measurement $\mathbf{z}_t$, or if saturated region skipping applies, no updates are needed. Otherwise, we continue by checking if the approximation error at the partition's current resolution is acceptable. If this is the case, we evaluate the inverse measurement model $s(m_{\mathbf{x}'}|\mathbf{z}_t)$ at the partition's

center and integrate the update into the map. If none of the previous criteria were met, a higher resolution is needed and the recursive function is called for each of the octree node's sub-divisions (octants). In practice, we also compress the measurement update using the wavelet transform and need to traverse the map's data structure. Both of these operations can efficiently be interleaved with the recursive adaptive sampling procedure. Note that although the presented algorithm is recursive, great flexibility exists for its implementation. For example, since each Haar scaling function only overlaps with its parent and children, all partitions at a given resolution and their descendants can be updated in parallel.

3.5 Experiments

We evaluate our approach on three different datasets, featuring depth cameras and LiDARs, in indoor as well as outdoor environments. Comparisons are presented to three state-of-the-art volumetric mapping frameworks: octomap [4], voxblox [29], and supereight2 [38]. Octomap and supereight2 are both used in multi-resolution occupancy mapping-mode. Voxblox only supports TSDFs mapping and is configured to use its default 'fast' integration method. In terms of implementation details, all approaches are evaluated using their publicly available reference implementations[3][4][5] and wrapped with the same code to process the training data.

For each dataset, we split the original data into training and test sets by reserving every 20th observation for testing and use the remaining frames for mapping. Test points are generated by sampling points along all rays in each test observation, with points along the beam being in free space and the endpoint being occupied. To obtain insights into the behavior of the different methods in various scenarios we compute the distance of each free-space test point to the closest surface point. This allows us to evaluate the performance in different range bands, including: i) small negative values assessing the ability to capture thin objects, ii) distances close to zero to evaluate the surface reconstruction quality, and iii) larger distances to obstacles to detect possible biases or approximation errors. This approach also avoids diluting a small number of challenging situations with a large number of easy-to-classify free space observations.

For each experiment, we report the overall Area under the ROC Curve (AUC) as a general indicator of classification performance. By integrating the Receiver Operating Characteristic (ROC) curve, the AUC quantifies how well each classifier discriminates

[3] https://github.com/OctoMap/octomap_mapping
[4] https://bitbucket.org/smartroboticslab/supereight2
[5] https://github.com/ethz-asl/voxblox

Algorithm 1: Wavemap recursive update

Input: Current measurement $\mathbf{z}_t$,
Previous map posterior $p(m \mid \mathbf{z}_{1:t-1})$,
Lower log-odds threshold $\mathcal{L}_{\text{min}}$,
Approximation error threshold ϵ_{thresh},
Maximum resolution res_{max},
Octree's root partition $\mathcal{V}_{\text{root}}$
Output: Updated map posterior $p(m \mid \mathbf{z}_{1:t})$

Function `RecursiveAdaptiveUpdate`$(\mathcal{V}, \mathbf{z}_t)$ **is**
  `// Use Eq.3.11 to skip partitions`
  update_type $\leftarrow$ `UpdateType`$(\mathcal{V}, \mathbf{z}_t)$
  if update_type $=$ FullyUnobserved **then**
    return
  end
  if (update_type $=$ FreeOrUnobserved
    and $\mathcal{L}_p(m_{\mathcal{V}} \mid z_{1:t-1}) \leq \mathcal{L}_{min}$) **then**
    return
  end
  `// Use Eq.3.13 to terminate early`
  $\epsilon_{\text{max}}(\mathcal{V}) \leftarrow$ `ApproximationError`$(\mathcal{V}, \mathbf{z}_t)$
  if ($\mathcal{V}_{res} = res_{max}$ **or** $\epsilon_{max}(\mathcal{V}) < \epsilon_{thresh}$) **then**
    $\mathcal{L}_p(m_{\mathcal{V}} \mid \mathbf{z}_{1:t}) \leftarrow \mathcal{L}_p(m_{\mathcal{V}} \mid \mathbf{z}_{1:t-1})$
    $+\mathcal{L}_s(m_{\mathcal{V}} \mid \mathbf{z}_t)$
    return
  end
  `// Otherwise, increase resolution`
  for $\mathcal{V}_{child} \in \mathcal{V}$ **do**
    `RecursiveAdaptiveUpdate`$(\mathcal{V}_{\text{child}}, \mathbf{z}_t)$
  end
end
`// Initialize map and start recursion`
$p(m \mid \mathbf{z}_{1:t}) \leftarrow p(m \mid \mathbf{z}_{1:t-1})$
`RecursiveAdaptiveUpdate`$(\mathcal{V}_{\text{root}}, \mathbf{z}_t)$

Table 3.1: Area Under the ROC Curve results for both datasets. Higher is better. The corresponding resource usages are in table 3.2.

Dataset	Res	octomap	super-eight2	voxblox	*ours (rays)*	*ours (beams)*
Panoptic	5cm	0.95	0.93	**0.99**	**0.99**	**0.99**
Flat	2cm	0.99	0.95	**1.00**	**1.00**	**1.00**
Newer	20cm	0.82	0.87	**0.92**	0.91	0.91
College	5cm	0.90	0.89	**0.97**	0.94	**0.97**

free and occupied space regardless of the classification threshold. We also report the classification accuracy for the individual range bands. Note that different accuracies can be obtained based on the chosen classification threshold. For this study, we set the thresholds for each framework on each dataset to the value that maximizes the difference between the True Positive Rate (TPR) and the False Positive Rate (FPR), weighed equally.

3.5.1 Accuracy evaluations

Panoptic mapping dataset

The first set of experiments is conducted using “Run 1" of the panoptic mapping dataset [48], which features depth camera recordings of a simulated studio apartment including realistic household objects. Octomap and voxblox do not support depth images directly, and hence the dataset’s images were first converted to pointclouds using the pinhole projection model used by both supereight2 and our method. The camera poses were obtained from the ground truth.

From the AUC values shown in Table 3.1 we can see that, when using larger cell sizes, only our proposed method can compete with voxblox. Being TSDF-based, voxblox can more accurately reconstruct smooth surfaces which account for large parts of the environment, giving it a distinct advantage. The remaining two methods have worse overall performance. When moving to a higher resolution the difference shrinks and all methods perform comparably. Looking at the results shown in Figure 3.3 we can clearly see where octomap and supereight2 accumulate their errors in the $5\,\mathrm{cm}$ resolution case. Octomap struggles to properly localize the surface boundary, while supereight2 is overly pessimistic, labeling cells far from the surface as occupied. Finally, one can see the trade-off between our method, using a beam-based model,

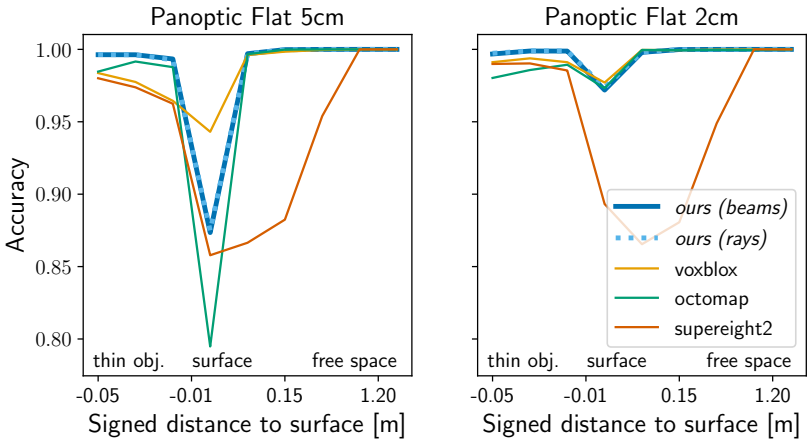


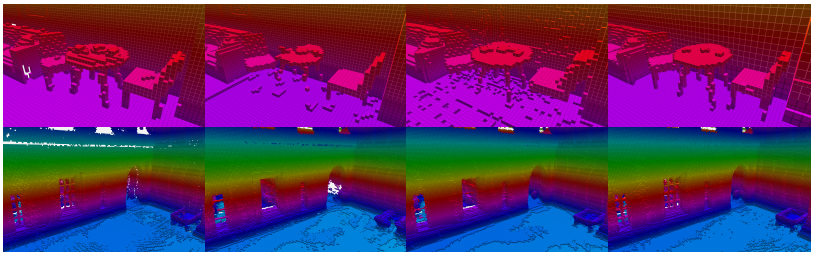
Figure 3.3: Accuracy in function of the distance to the surface on the Panoptic mapping dataset at different resolutions. Higher is better.

and voxblox, using a TSDF model. Voxblox has better at the surface reconstruction performance while our approach is better at reconstructing thin objects. This difference can also be seen in Figure 3.4 where the chair is missing its legs in the voxblox reconstruction. Looking at the $2\,\mathrm{cm}$ resolution case, all methods but supereight2 perform almost identically. Supereight2 still produces pessimistic results, which likely stem from the approximations used to achieve its impressive speed.

Newer College dataset

The second set of experiments uses the Cloister sequence from Collection 2 in the Newer College dataset [49]. This sequence was chosen because it captures geometry with a wide range of scales including wide-open outdoor spaces, indoor spaces with arches and sculptures, and vegetation. Odometry estimates and undistorted point clouds were obtained using FastLIO2 [50] processing the Ouster OS0-128 IMU and point cloud data. The motion-compensated point clouds were used for all frameworks except supereight2, which operates using dense range images and does not yet support motion-undistortion.

Looking at the AUC numbers in Table 3.1 we immediately see that this real-world



octomap supereight2 voxblox ours (beams)

Figure 3.4: Qualitative reconstruction comparisons featuring detailed geometry on scenes of the Panoptic mapping (top) and Newer College (bottom) datasets, both at 5cm resolution.

LiDAR dataset is more challenging than the previous synthetic one. When using a coarse $20\,\mathrm{cm}$ resolution octomap performs the worst, with voxblox and our method achieving the best results, and supereight2 landing in the middle. Moving to a higher resolution of $5\,\mathrm{cm}$ octomap and supereight2 end up performing similar while voxblox slightly outperforms our approach. However, the detailed results shown in Figure 3.5 reveal interesting insights. At $20\,\mathrm{cm}$ resolution octomap struggles to produce accurate surfaces. We also see that our approach and supereight2 have similar performance when it comes to reconstructing the surface but our approach performs slightly better when classifying free space in the vicinity of obstacles. Voxblox again performs the best in surface reconstruction and free space classification, but suffers in the thin object reconstruction domain. Moving to a finer $5\,\mathrm{cm}$ resolution the change is similar to that observed in the Panoptic dataset. The accuracy of every method improves and they move closer together, with supereight2 failing to accurately predict free space close to surfaces. The differences between the other three methods are characterized by octomap not reconstructing thin objects accurately while both voxblox and ours (beams) perform equally well.

Sensor model ablation

To verify the benefit of the more costly uncertainty aware sensor model proposed in Section 3.4.2, we conduct an ablation comparing our proposed sensor model, *ours (beams)*, with one that disregards angular uncertainty, *ours (rays)*. As the Panoptic Flat dataset contains no noise on observation or pose there is, as to be expected, no

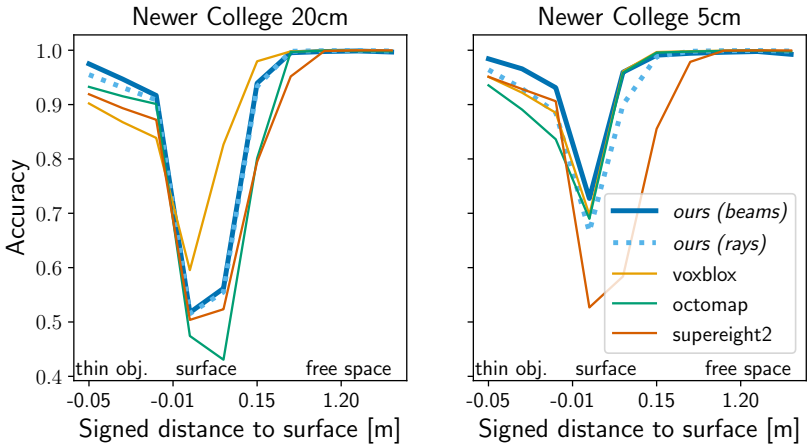


Figure 3.5: Accuracy in function of the distance to the surface on the Newer College dataset at different resolutions. Higher is better.

difference between the two models. On the Newer College dataset, however, there are visible differences. In the coarse setting the proposed uncertainty-aware model improves the ability to reconstruct thin objects. Moving to the higher resolution case both the ability to reconstruct surfaces and thin objects are significantly improved by our proposed model.

These accuracy evaluations showed several things. The proposed method *ours (beams)* compares favorably to the other three methods. Despite the natural advantage voxblox has in surface reconstruction tasks, being a TSDF-based method, our approach performs on par while having superior performance in thin object reconstruction. The uncertainty-aware sensor model also improves the quality of the map close to surfaces and when dealing with thin surfaces, allowing the reconstruction of objects that other methods can't capture when using the same cell size.

3.5.2 Efficiency evaluations

We evaluate the memory usage as well as the runtime of our method in comparison to the three baseline methods. Memory usage is reported as the amount of RAM used by the method as well as the memory used by the map data structure. While

our framework can be implemented using various data structures, we used octomap's octree implementation to keep the comparison as fair as possible. The runtime is reported as the elapsed wall time and the cumulative CPU time across all threads, allowing a fair comparison between single-threaded and multi-threaded methods. All frameworks have their visualizations disabled and all experiments are performed on the same desktop computer with an Intel i9-9900K CPU.

From the numbers shown in Table 3.2 we can see that supereight2 ranks first in terms of wall time on the depth camera dataset, and second best for LiDAR. However, the memory usage of its maps is relatively large owing to the fact that it estimates occupancy using weighted averaging instead of log-odds updates (requiring 2 floats per cell instead of 1) and focuses its implementation primarily on speed. Voxblox, as to be expected from a TSDF-based method, has the largest map sizes at higher resolutions but is computationally efficient. Octomap produces large maps, in comparison to our method, and is the slowest of all compared methods by an order of magnitude. Octomap's significant slowdown at high resolutions is caused by the fine-to-coarse model employed by their integrator which needs to touch every single cell. Our proposed method obtains maps that are significantly smaller than those of octomap despite using the same underlying data structure.

The runtime of our method, when looking at the CPU time, is equal or better than that of supereight2. However, as supereight2's implementation uses multiple threads the real-world performance of it is still better. The difference in runtime and memory usage between our method and octomap clearly shows the benefits of using wavelets to represent the map as it enables good compression and allows the use of an efficient coarse-to-fine integrator capable of skipping unnecessary work.

Comparing the memory and runtime of *ours (rays)* and *ours (beams)* we can see that the price for the improved quality is larger maps by about 30% to 70% depending on the resolution and an increase in runtime of around 50%. These increases stem from the fact that the uncertainty-aware model needs to update more voxels and that the map contains more fine details and voxels with partial occupancy values. Overall, our proposed method shows good general performance in both memory usage and runtime, with clear avenues for improvements. The wall time could be reduced significantly using multi-threading, which is easily achievable due to the independence of the voxel updates. Moreover, we believe the memory used to store the map itself could be reduced further by using a more efficient data structure such as the one proposed by OpenVDB [47]. These extensions will be added to the open-source code.

Table 3.2: Computational resource usage at different resolutions. Lower is better.

			Memory (MB)		Time (s)	
Dataset	Res	Framework	RAM	Map only	CPU time	Wall time
Panop. Flat	5cm	octomap	162.35	6.50	130.32	129.00
		supereight2	158.23	46.09	27.79	**4.76**
		voxblox	229.96	36.90	58.58	10.68
		ours (rays)	135.69	**4.17**	**5.58**	6.78
		ours (beams)	**130.04**	5.65	6.94	7.20
	2cm	octomap	6202.39	50.94	773.16	763.39
		supereight2	448.38	285.07	50.83	**9.32**
		voxblox	663.53	348.15	244.69	24.61
		ours (rays)	343.26	**39.09**	**33.00**	34.80
		ours (beams)	**294.58**	67.81	57.56	57.51
Newer Coll.	20cm	octomap	203.25	20.78	688.71	709.99
		supereight2	249.03	107.79	411.67	67.14
		voxblox	261.02	66.32	228.12	**48.07**
		ours (rays)	180.86	**6.94**	**87.39**	88.78
		ours (beams)	**138.92**	8.82	107.67	113.26
	5cm	octomap	14404.76	981.02	36252.70	35790.60
		supereight2	2926.42	2333.93	2853.12	404.19
		voxblox	3718.85	2362.58	1788.90	**162.36**
		ours (rays)	1192.95	**241.84**	**1656.26**	1671.58
		ours (beams)	**1065.21**	402.18	2085.05	2083.61

3.5.3 Multi-sensor multi-resolution mapping

One key advantage of our framework is its natural ability to integrate multiple sensors with different settings. In this experiment, we show qualitative results of our mapping framework running in multi-sensor mode on the DARPA SubT Finals dataset [51]. In Figure 3.6, we show the output of our framework simultaneously integrating two Robosense Bpearl dome-LiDAR sensors and one Velodyne VLP-16 LiDAR. The Bpearls were angled to scan the ground around the robot, while the VLP-16 was mounted horizontally to provide long-range observations. As the sensors provide information for different purposes, we integrate them with different resolutions into the map. The VLP-16, responsible for long-range mapping and exploration, is integrated up to a maximum range of $30\,\mathrm{m}$ and a maximum resolution of $16\,\mathrm{cm}$. At the same time, the Bpearls, responsible for local terrain mapping to enable navigation of a

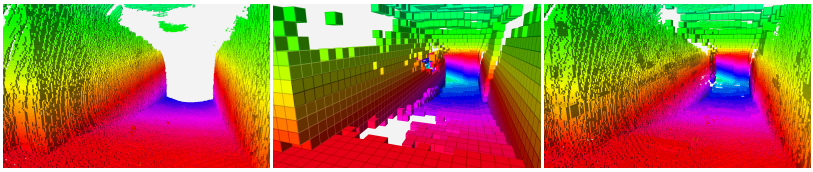


Figure 3.6: Example of our framework performing multi-sensor, multi-resolution volumetric mapping on the DARPA SubT Finals dataset, combining data from 2 ground-facing LiDARs at 2cm resolution (left) and 1 horizontal LiDAR at 16cm resolution up to a range of 30m (center) into a single map (right).

quadruped, are integrated up to a resolution of $2\,\mathrm{cm}$ and range of $2.5\mathrm{m}$. This results in a unified map that supports foot placement, local trajectory and global exploration planning without wasting resources on high-resolution map reconstruction in areas where it is not needed. While shown here for multiple LiDAR sensors, the same approach has also been used for mobile manipulation setups using a 3D LiDAR for navigation and RGB-D cameras for scene reconstruction, resulting in a map that supports navigation as well as fine-grained manipulation.

Note that existing frameworks, such as UFOMap [39] and supereight2 [38], could also be extended to support multi-sensor, multi-resolution mapping. The key difference is that they cannot do this efficiently since the resolution levels in their maps require explicit synchronization. The associated overhead is often reduced by performing the synchronization in a lazy fashion. However, this optimization is ineffective when regions of the map are concurrently updated at multiple resolutions, which requires continuous synchronization. Our framework does not suffer from this limitation because wavelet decompositions synchronize their resolution levels implicitly. A further advantage of wavelet encoding is that when a region is updated at a lower resolution, the region's high-resolution cells remain consistent without requiring any additional processing. In other words, the computational complexity of our framework purely scales with the update resolution, regardless of each region's effective maximum resolution. In practice, this makes it possible to seamlessly and efficiently fuse long-range, lower-resolution sensors such as LiDARs with local, high-resolution sensors such as depth cameras.

3.6 Conclusion

In this chapter, we introduced *wavemap*, a hierarchical volumetric mapping framework inspired by multi-resolution analysis. The MRA theory guarantees that using wavelet decomposition, we can safely and very efficiently integrate new observations in a coarse-to-fine manner. The resulting gains in computational efficiency, together with early stopping criteria for the integrator, allow us to use more complex sensor models such as the proposed angular uncertainty-aware model. In experiments on synthetic RGB-D and real-world 3D LiDAR data, we demonstrate that our proposed method achieves high-quality results while being efficient in terms of memory and compute requirements. We also demonstrate how our method can incorporate observations from multiple sensors into a single map with per-sensor resolution. This allows the use of a single map representation for tasks that would have required several dedicated maps in the past. Finally, we open source the implementation of our approach to facilitate future research.

Chapter 4

Multi-resolution collision avoidance

Fast and reliable obstacle avoidance is an important task for mobile robots. In this chapter, we propose an efficient reactive system that provides high-quality obstacle avoidance while running at hundreds of hertz with minimal resource usage. Our approach combines wavemap, a hierarchical volumetric map representation, with a novel hierarchical and parallelizable obstacle avoidance algorithm formulated through Riemannian Motion Policies (RMP). Leveraging multi-resolution obstacle avoidance policies, the proposed navigation system facilitates precise, low-latency (36ms), and extremely efficient obstacle avoidance with a very large perceptive radius (30m). We perform extensive statistical evaluations on indoor and outdoor maps, verifying that the proposed system compares favorably to fixed-resolution RMP variants and CHOMP. Finally, the RMP formulation allows the seamless fusion of obstacle avoidance with additional objectives, such as goal-seeking, to obtain a fully-fledged navigation system that is versatile and robust. We deploy the system on a Micro Aerial Vehicle and show how it navigates through an indoor obstacle course. Our complete implementation, called *waverider*, is made available as open-source[1].

The work described in this chapter is presented in the following publication:

- V. Reijgwart*, M. Pantic*, R. Siegwart, L. Ott, "Waverider: Leveraging Hierarchical, Multi-Resolution Maps for Efficient and Reactive Obstacle Avoidance," ICRA 2024

[1] https://github.com/ethz-asl/waverider

4.1 Introduction

Reactive, precise, and reliable obstacle avoidance is vital for mobile robots to safely and efficiently navigate through changing or partially unknown environments. Since obstacle avoidance is an always-on process, it must use minimal computational resources and seamlessly integrate with the robot's other tasks. Existing approaches range from simple reactive methods using 1D distance sensors to optimization-based systems requiring complete 3D maps and vary in complexity, reaction time, and obstacle resolution.

While collision avoidance systems that operate directly on raw sensor data may exhibit exceptionally low latency, they can only guarantee safety with respect to consistently observed obstacles within the Field of View (e.g. [52]). One way to introduce memory without losing generality is to use volumetric maps. They can model obstacles of arbitrary shape and explicitly distinguish free and unobserved space. Volumetric maps are well suited to ensure safety even in unknown environments. However, fixed-resolution volumetric mapping frameworks tend to suffer from excessive memory overheads and latency. These can be overcome by using hierarchical volumetric representations such as octomap [4], UFOMap [39], supereight [36], or wavemap [7]. While several works investigated the use of hierarchical maps for global path planning, most collision avoidance systems still process all obstacles at the highest resolution. Yet, intuitively, one would expect that distant obstacles could be considered at a lower resolution than nearby ones without significantly affecting the robot's behavior.

We use RMPs [53] to formulate a navigation algorithm that is inherently multi-scale and hierarchical. RMPs are purely reactive in nature, and as such, can be formulated extremely efficiently and executed with low latency at controller frequency. Other sampling- or optimization-based methods often need pre- and post-processing steps such as the generation of an ESDF or trajectory smoothing. Conversely, RMPs are formulated as second-order dynamical systems and directly output accelerations, which typically leads to gradual changes and smooth paths. RMPs have some similarities to the well-known potential fields [54], but are a much more expressive framework due to the inclusion of the Riemannian metric that modulates each policy's strength and directionality.

In this chapter, we develop a reactive and safe obstacle avoidance method using RMPs [53] that is tailored to hierarchical volumetric map representations. We numerically analyze the effects of obstacle resolution on the policy's approximation error as a function of the distance between the robot and the policy. Based on this analysis, we derive a function that computes the ideal resolution for querying the

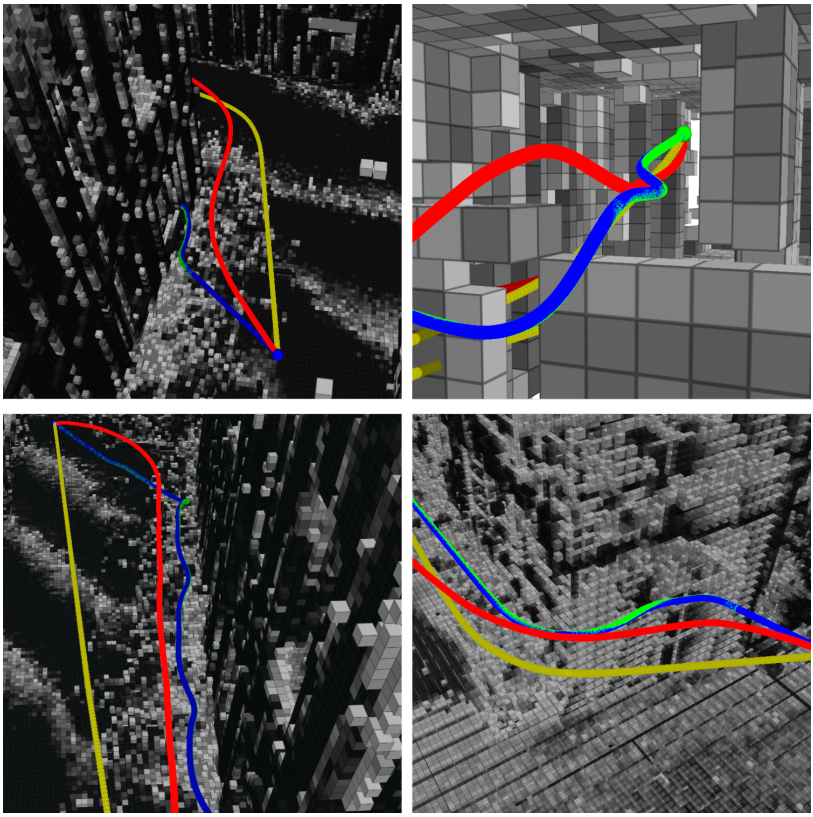


Figure 4.1: Example trajectories comparing our multi-resolution collision avoidance method (red) to equivalent RMP-based formulations that consider all obstacles at the highest resolution within a radius of 1 m (green) and 3 m (blue). The fixed-resolution RMP trajectories are jerkier and more prone to get stuck (top-left). CHOMP (brown) yields smooth, albeit overly cautious trajectories and occasionally cuts through obstacles (top-right, bottom-left).

map at a given distance from the robot – allowing us to balance computational effort and accuracy. Using this function, we develop an algorithm that efficiently generates multi-resolution avoidance policies from a hierarchical map.

The contributions of this chapter are:

- An efficient hierarchical obstacle policy generation algorithm;
- Numerical analysis of the approximation error induced by hierarchical navigation policies;

The correctness of the numerical analysis is statistically validated through a large number of experiments in simulation. Extensive comparisons with baselines and CHOMP [8] demonstrate the favorable runtime and efficiency of our method. Finally, we demonstrate real-world applicability by deploying our system onboard an MAV running at 200 Hz.

4.2 Related Work

A core decision in any obstacle avoidance system is the environment representation. State-of-the-art systems combine a volumetric map such as a truncated signed distance field [10, 55] or an octree-based occupancy map [4, 7, 36, 39] with either a search-based method such as A*[56], a sampling-based approach such as RRT [57], or an optimizer such as CHOMP [8, 58]. All of these methods are comparably slow, as the mapping-planning cycle has multiple performance bottlenecks, and the sampling or optimization steps often rely on post-processed maps. Recently, end-to-end learning-based methods were shown to be effective for collision avoidance [59]. However, their data-driven nature still comes with a lack of generalizability across different environments, sensors, and robot dynamics.

Reactive approaches that operate directly on volumetric maps or even raw LiDAR data exist [60], but these methods have considerable memory and computing requirements due to their dense data representation. Although hierarchical volumetric maps have received considerable attention from the planning community, most works focused on global planning [38, 61, 62]. Multi-resolution anytime planners [63] have been proposed that bridge the gap to local planning. However, their global context makes achieving the update rates required for low-latency reactive collision avoidance challenging in 3D. Funk et al. [64] propose a full planning pipeline that leverages multi-resolution for efficient orientation-aware planning in environments with very narrow openings. However, their evaluations are performed on pre-computed static

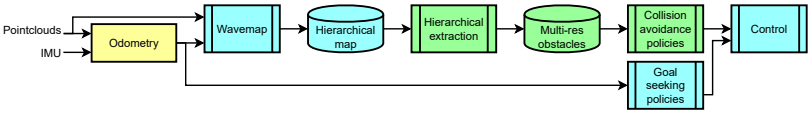


Figure 4.2: Block diagram of the proposed navigation system. External components are highlighted in yellow, tightly integrated components in blue, and new components introduced in this chapter in green.

maps without perception in the loop, which makes it difficult to judge the system's latency in a reactive collision avoidance setting. Closest to our work is the hierarchical collision avoidance system presented by Goel et al. [65] that adapts the map resolution based on the motion primitives considered by the planner. The method is used in a teleoperation setting and shows promising results in simulated and real environments. However, a significant part of the system's efficiency results from using a bespoke, purely local map representation whose resolution is set by the planner, which is harder to reuse for additional tasks, including global planning. In comparison, our system achieves comparable efficiency levels using generic hierarchical occupancy maps. This is explained by the efficiency of RMPs, and the fact that our method does not rely on expensive ESDFs. One final benefit of our proposed architecture, compared to both [64, 65], is its high degree of modularity. Formulating obstacle avoidance as a motion policy makes it easy to combine with other policies representing additional objectives such as goal-seeking, visual servoing, or aerial manipulation.

4.3 Method

In the following sections, we describe our approach to efficiently extract multi-resolution obstacle avoidance policies from hierarchical maps and how they integrate with high-level task policies. Figure 4.2 shows the main parts of the system, consisting of: 1) a volumetric, hierarchical map representation (Section 4.3.1), 2) an algorithm for obstacle extraction (Section 4.3.2), 3) an RMP-based reactive navigation system (Section 4.3.4). For each obstacle cell extracted in 2) an individual obstacle avoidance policy is generated (Section 4.3.3), and combined with all other policies through the RMP framework.

4.3.1 Hierarchical map

The proposed method is compatible with any hierarchical occupancy mapping framework, e.g. [4, 36, 39]. We chose to use wavemap [7], as it simultaneously achieves state-of-the-art accuracy, memory, and computational efficiency. In a similar fashion to other methods, wavemap leverages octrees to achieve this efficiency. However, instead of storing absolute occupancy values, each node stores Haar wavelet coefficients. Using wavelets achieves significant compression and, more importantly, guarantees that all resolution levels are implicitly synchronized and always up to date. An efficient coarse-to-fine measurement integration algorithm allows wavemap to integrate depth measurements with low latency, even on computationally constrained platforms.

4.3.2 Obstacle cell extraction

As will be substantiated in Section 4.4, reducing the resolution of obstacles as the distance to the robot increases does not introduce significant approximation errors. By representing obstacles at the appropriate resolution, it is therefore possible to efficiently consider fine nearby obstacles and the broader spatial context simultaneously. In this section, we present a hierarchical algorithm that efficiently gathers multi-resolution obstacles by traversing the map in a coarse-to-fine manner. The algorithm (Algorithm 2) starts at the lowest resolution level (root node) of the map and recursively visits each node's higher-resolution children. The algorithm stops expanding a node when that node either has no children or its distance d to the robot exceeds $d_{max}(\lambda)$. We use $d_{max}(\lambda) = 3^{\lambda/3} - 0.25$ where λ corresponds to the node's height in the octree[2]. Once such a terminal node is found, an obstacle cell is created if the node or any of its children is occupied. Figure 4.4 visualizes $d_{max}(\lambda)$ and the resulting maximum distance up to which obstacles are included.

4.3.3 Collision avoidance policy generation

For each obstacle cell returned by the previously described algorithm, an individual obstacle-avoidance policy $\mathcal{P}$ [53] is created. In the following, we give a short summary of the most important aspects of motion planning using RMP, however for more details and complete formulas of helper functions we refer to the original text [53]. A policy $\mathcal{P}$ consists of an acceleration function $\ddot{\mathbf{x}} = f(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}^3$ and a Riemannian metric $\mathbf{A}(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}^{3\times3}$, where $\mathbf{x} \in \mathbb{R}^3$ refers to the robot's current position. The function f drives the robot according to the policy, while the Riemannian metric $\mathbf{A}$ defines a

[2] A height of 0 corresponds to the highest resolution/smallest voxel size.

Algorithm 2: Hierarchical obstacle extractor

Input: Hierarchical occupancy map $\mathcal{M}$,
Robot position $\mathbf{p}$
Output: Set of multi-resolution obstacles $\mathcal{O}$

Function `RecursiveExtractor`$(\mathcal{V}, \mathbf{p})$ **is**
  $d \leftarrow ||\mathcal{V}_{\text{center}} - \mathbf{p}||_2$
  if $d_{max}(\mathcal{V}_\lambda) < d$ **then**
    if `IsOcc`$(\mathcal{V})$ **or** `HasOccChild`$(\mathcal{V})$ **then**
      $\mathcal{O}.insert(\mathcal{V})$
    end
    return
  end
  if not `HasOccChild`$(\mathcal{V})$ **then**
    return
  end
  for $\mathcal{V}_{child} \in \mathcal{V}$ **do**
    `RecursiveExtractor`$(\mathcal{V}_{\text{child}}, \mathbf{p})$
  end
end
`// Initialize and start recursion`
$\mathcal{V}_{\text{root}} \leftarrow$ `GetOctreeRoot`$(\mathcal{M})$
$\mathcal{O} \leftarrow$ `RecursiveExtractor`$(\mathcal{V}_{\text{root}}, \mathbf{p})$

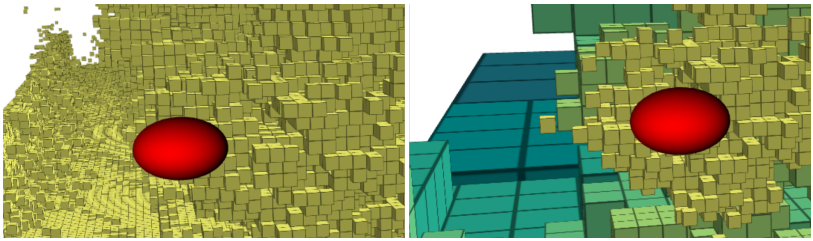


Figure 4.3: Comparison of an environment represented using fixed-resolution (left) and hierarchical obstacle cells (right). Our approach uses hierarchical cells, whose resolution (light brown to dark green) is high close to the robot (red) and decreases with distance.

(possibly directional or isotropic) weight of the policy in comparison to other policies. Following [53], multiple policies $\{\mathcal{P}_0, \ldots, \mathcal{P}_N\}$ can be summed into an equivalent policy $\mathcal{P}_C$ using

$$\mathcal{P}_c = (\mathbf{f}_c, \mathbf{A}_c) = \left(\left(\sum_i \mathbf{A}_i \right)^+ \sum_i \mathbf{A}_i \mathbf{f}_i, \sum_i \mathbf{A}_i \right). \tag{4.1}$$

We use the obstacle avoidance repulsor from [53] as a policy template for each found obstacle cell. It is formulated as a combinationof a pure repulsor $\mathbf{f}_{rep}$, a velocity-dependent damper $\mathbf{f}_{damp}$, and a metric (weight) that becomes 0 if the robot's velocity does not point towards the obstacle. The repulsor is defined as

$$\mathbf{f}_{rep}(\mathbf{x},\, \mathbf{r},\, d) = \eta_{rep} \exp\left(-\frac{d}{\upsilon_{rep}} \right) \mathbf{r}\,, \tag{4.2}$$

where d is the distance to the obstacle, $\mathbf{r}$ is the unit vector pointing from the obstacle to the robot, and η_{rep} and υ_{rep} are tuning parameters to set the repulsor strength (η_{rep}) and scaling (υ_{rep}). Similarly, the damper is defined as

$$\mathbf{f}_{damp}(\dot{\mathbf{x}},\, \mathbf{r},\, d) = \eta_{damp} \Big/ \left(\frac{d}{\upsilon_{damp}} + \epsilon \right) \cdot \mathbf{P}_{obs}(\dot{\mathbf{x}},\, \mathbf{r})\,, \tag{4.3}$$

again with η_{damp} as a strength parameter and υ_{damp} as a scaling parameter. ϵ is a sufficiently small constant to ensure numerical stability. $\mathbf{P}_{obs}(\dot{\mathbf{x}},\, \mathbf{r})$ projects the robot velocity onto the direction vector pointing from the obstacle to the robot and captures how much the robot moves towards the obstacle. Finally, the full obstacle avoidance policy is defined as the tuple $\mathcal{P}_{obs} = (\mathbf{f}_{obs}, \mathbf{A}_{obs})$:

$$\mathbf{f}_{obs}(\mathbf{x},\, \dot{\mathbf{x}},\, \mathbf{r},\, d) = \mathbf{f}_{rep}(\mathbf{x},\, \mathbf{r},\, d) - \mathbf{f}_{damp}(\dot{\mathbf{x}},\, \mathbf{r},\, d) \tag{4.4}$$

$$\mathbf{A}_{obs}(\mathbf{x},\, \dot{\mathbf{x}},\, \mathbf{r},\, d) = w_r(d,) \cdot \boldsymbol{s}(\mathbf{f}_{obs})\, \boldsymbol{s}(\mathbf{f}_{obs})^T\,. \tag{4.5}$$

$\boldsymbol{s}(\cdot)$ is a soft-normalization function. Please refer to [53] for the detailed formulations of $\mathbf{P}_{obs}$ (eq. 68) and $\boldsymbol{s}$ (eq. 24). w_r scales the policy response based on a distance parameter r, which influences the policy's maximum active range according to $w_r(d) = \frac{1}{r^2} d^2 - \frac{2}{r} d + 1$. For each of the thousands of found obstacle cells such a policy is created. All cells at the same scale level λ are then summed according to Eq. (4.1), and all resulting combined policies of all scales are then again summed using Eq. (4.1). The scale level λ is used to set the RMP's parameters as follows: $\upsilon_{damp} = 0.45\lambda$, $\upsilon_{rep} = 0.75\lambda$, and $r = 1.5\lambda$. Modulating υ_{damp}, υ_{rep}, and r, allows setting the sphere of influence of policies, and for example determines the

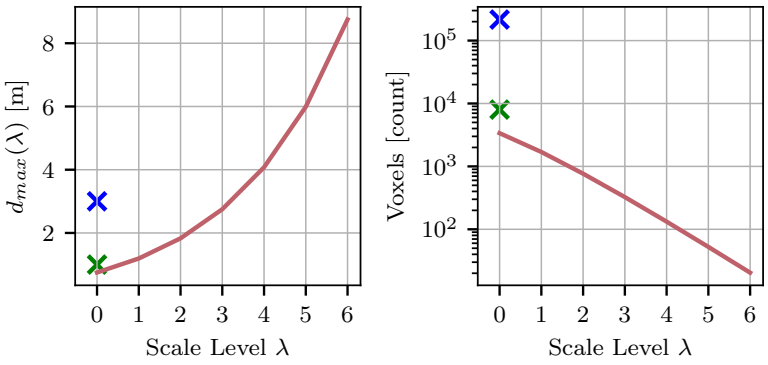


Figure 4.4: Left: Perceptive radius defined by $d_{max}(\lambda)$ as used in the obstacle filter (red). Limited, fixed-resolution comparison variants used in 4.5.1 are marked with a blue resp. green cross. Right: Worst-case counts of voxels to visit. Even with small perceptive radii, the fixed-resolution variants need to potentially iterate over significantly more voxels to provide the same quality of obstacle avoidance (log scale).

traversability of narrow corridors. By using the tuning proposed above, coarse obstacles naturally have a larger sphere of influence. The distance and size of the obstacle cell are used to scale the policy's Riemannian metric, which can be interpreted as a multi-dimensional weight and modulates the policy's strength and activation radius. The Riemannian metric ensures that the relative direction to the obstacle cell is taken into account such that there is only a repulsion component if the robot's velocity points towards this obstacle. In Figure 4.3, examples of obstacle cells are shown for both uniform and hierarchical cell generation.

4.3.4 Navigation system integration

We use the simple goal-attractor policy described in [53] to combine the previously described summation of obstacle avoidance policies with goal-seeking behavior. The goal-attractor policy is defined as:

$$\begin{aligned} \mathbf{f}_a(\mathbf{x}, \dot{\mathbf{x}}) &= \alpha_a \boldsymbol{s} \left(\mathbf{x}_a - \mathbf{x}\right) - \beta_a \dot{\mathbf{x}} \\ \mathbf{A}_a(\mathbf{x}, \dot{\mathbf{x}}) &= \mathbb{I}^{3\times 3} \end{aligned}, \tag{4.6}$$

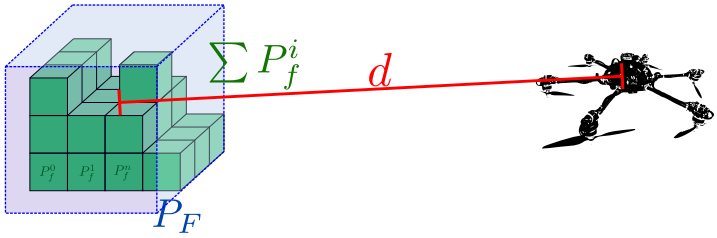


Figure 4.5: Example of obstacles that can be either modeled by a single, large policy (P_F) or multiple small, high-resolution policies (P_f^i). The distance d represents the distance from the robot to the center of the obstacle block.

where $\alpha_a, \beta_a > 0$ are tuning parameters, and $\mathbf{x}_a$ is the desired goal location. In each iteration, all policies are evaluated, summed up, and the resulting acceleration executed on the robot. For simulation experiments, the policies are run as fast as possible, whereas during field tests the policies are evaluated at the robot's control frequency (200 Hz). Note that it is straightforward to replace or combine the goal-seeking policy with other policies addressing tasks such as visual servoing, terrain following, manipulation, or assisted manual control, as has been shown e.g. in [66].

4.4 Hierarchical Policy Approximation Error

Naturally, one wonders what the impact of incorporating distant obstacles at a reduced resolution is. In this section, we study the influence of replacing a sum of obstacle avoidance policies with a single policy at the center of such a block. In the obstacle cell extraction algorithm, the octree is traversed to a deeper or shallower level depending on the distance to the robot. This implies that at larger distances, fewer policies at slightly different locations contribute to the overall navigation result instead of a sum of many individual policies. In the following, we show what relative changes in policy outputs and quality these abstractions entail, using the toy example in Figure 4.5 for the analysis.

We conduct a numerical analysis to simulate the relative changes between the single simplified policy P_F and the granular, high-resolution set of policies $\sum P_f^i$ in both policy strength and directionality for three scenarios: 1) the toy example in Figure 4.5 (labeled "Fig" in Figure 4.6), 2) a random sampling of 16 occupied voxels, respectively their resulting policies ("R16"), and 3) a completely occupied block resulting in 64

policies ("All"). The same $4 \times 4 \times 4$ block with $10\,\mathrm{cm}$ voxels is used in all scenarios. As is visible in Figure 4.6, the induced errors are negligible both in angular error as well as relative strength (magnitude) of the resulting policies. As to be expected, errors are higher when the distance to the voxels is smaller. The combination of multiple policies at different scales provides the best compromise; it minimizes the number of policies needed while also providing low approximation error over the entire distance.

4.5 Experiments

We perform a comprehensive set of experiments to evaluate the navigation success rates, computational efficiency, and real-world applicability of the proposed system. To provide context, we include comparisons with CHOMP [8]. CHOMP generates complete trajectories and requires an Euclidean Distance Field (EDF), which is time-consuming to generate ($\approx 30\,\mathrm{s}$ for the maps used). By contrast, an RMP-based navigation framework is inherently reactive and only needs obstacle information which is readily available in the volumetric map.

4.5.1 Statistical evaluation and comparison

Despite the purely reactive nature of the proposed system, we are interested in its capability and performance in finding moderately complex trajectories in realistic maps. To this end, we perform an in-depth randomized evaluation on maps generated from the *Newer College* LiDAR dataset [49] with a min voxel size of $10\,\mathrm{cm}$. We use subsections of two maps – `mine` and `math`, visualized in Figure 4.7 – in which we sample random start and end points and let each navigation algorithm find a smooth, collision-free trajectory. We evaluate a total of four algorithms; a) the proposed hierarchical system as described in Section 4.3, b) an implementation of CHOMP [8], c) a non-hierarchical variant of our system that only uses the highest resolution voxels, up to a maximum distance of $1\,\mathrm{m}$, and d) $3\,\mathrm{m}$, respectively. The non-hierarchical variants serve to illustrate the effects of the reduced perceptive radius, which is limited due to the significantly higher computational costs inherent to single-resolution approaches at small voxel sizes. All reactive, RMP-based variants are used in an end-to-end fashion, meaning that the policies are repeatedly updated and integrated until the robot is at a stand-still, either at the goal or in a local minima. Obstacle cells are updated from the map whenever the displacement since the last update exceeds $0.05\,\mathrm{m}$. CHOMP is configured to run with $N = 500$ trajectory points until it converges ($\epsilon_{rel} < 1e^{-5}$) or a maximum iteration count (100) is reached.

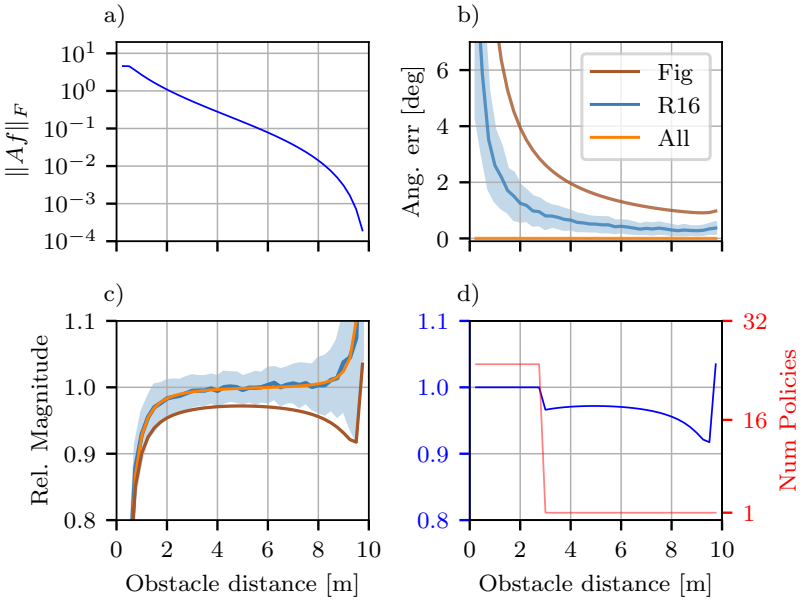


Figure 4.6: a) Plot of the typical absolute policy strength (log-plot) w.r.t. obstacle distance. Subfigure b) and c) visualize the angular and relative magnitude error of approximating the high-resolution policies with a single coarse approximation. 'Fig' shows this for the exact configuration seen in Fig. 4.5, 'R16' for a random selection of 16 occupied voxels at high resolution (thus the covariance), and 'All' for a fully occupied volume. d) Illustration of the approximation error for a hierarchical policy, where a full-resolution policy is used below $2.5\,\mathrm{m}$ distance and a single summary policy at larger distances. The spike in the approximation error's magnitude at a distance of $10\,\mathrm{m}$ is unimportant, as the absolute strength at this distance nears 0.

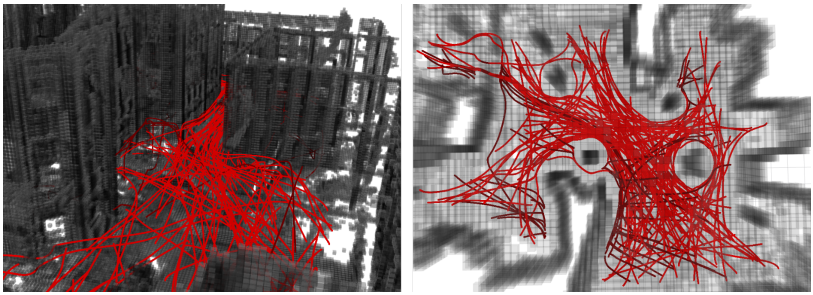


Figure 4.7: Qualitative visualization of the map scenarios used for statistical evaluation. Left shows a perspective rendering of the `math` scenario, right is a top-down rendering of the `mine` scenario. The red lines are example trajectories from our proposed navigation algorithm. Trajectories stuck in local minima are marked in dark red.

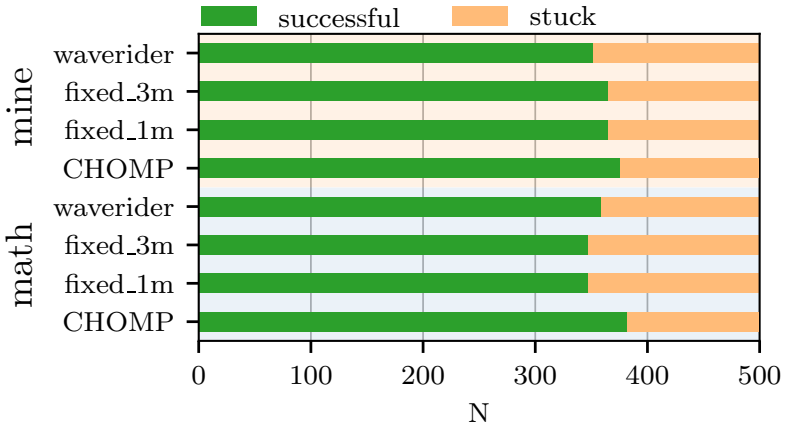


Figure 4.8: Success rates for all algorithms on both maps with 500 randomized trials each. CHOMP runs that did not terminate within the allocated time budget are labeled as stuck.

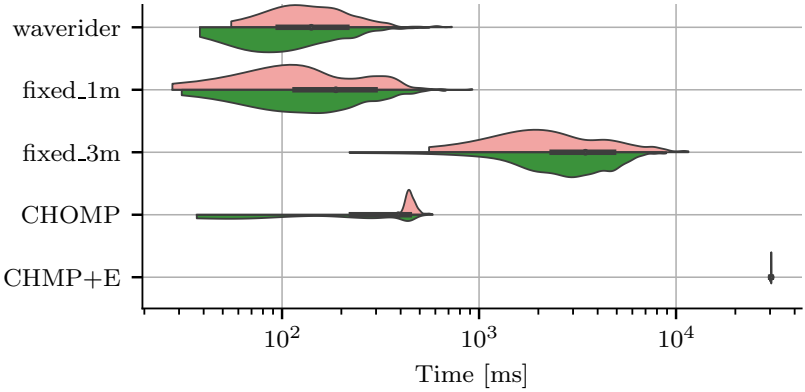


Figure 4.9: Timing distributions for rest-to-rest trajectories on map `math`. Green parts only include successful trajectories, red parts only stuck ones. CHOMP clearly shows increased calculation time for failing trajectories as it runs more solver iterations. Note the log scale and the drastically increased runtime for the fixed-resolution variant. For context, `CHMP+E` visualizes the cost of a trajectory, including the necessary collision distance (EDF) pre-processing for a map for CHOMP.

To demonstrate the relative performance of the proposed system, we provide a detailed look at planning success rate, planning time, and distances to obstacles. Figure 4.8 shows the relative amount of successfully found trajectories, i.e. that reach the goal location and do not get stuck. All algorithms perform similarly well and solve about 75% of all tasks, which is rather good considering that they are all local and not global planners. Due to their different nature, the reactive algorithms get (safely) stuck in local minima, whereas the optimization-based CHOMP method may simply not converge to a solution that is collision-free.

Figure 4.9 provides detailed statistics of the measured runtimes of the different algorithms. Noteworthy is the drastic increase in runtime with larger perceptive radii, which makes the use of large amounts of occupancy information intractable when a fixed-resolution representation is used. A major difference between our proposed method and CHOMP is its purely reactive nature. While we compare full rest-to-rest trajectory runtimes in Figure 4.9, in practice only a single iteration is calculated at every controller iteration. Effectively, this provides full obstacle avoidance navigation at a marginal compute cost – approximately 100 µs per step on average – and a few

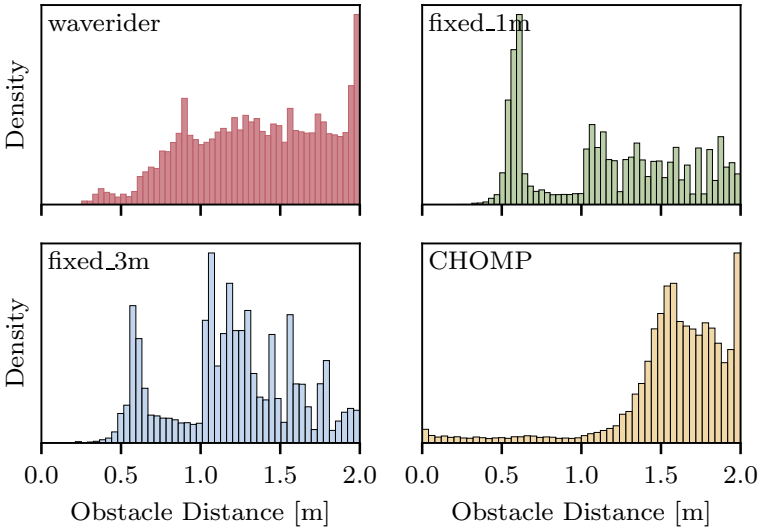


Figure 4.10: Histogramms of distance to obstacles over $50\,000$ random trajectory traces from each algorithm. The EDF used in the evaluation is truncated at $2\,\mathrm{m}$, with everything above that value being considered far away from obstacles.

milliseconds per step involving obstacle updates. Conversely, CHOMP only provides results after full convergence. To provide insights into trajectory safety, we evaluate the distance to the closest occupied obstacle for each step along each evaluated trajectory from the randomized tests. The resulting distributions are visualized as histograms in Figure 4.10. The proposed hierarchical approach shows a safe distribution with no parts of the trajectories getting close to obstacles. The two fixed-resolution algorithms frequently travel *much* closer to obstacles due to their limited perceptive fields, whereas CHOMP may output unsafe states in case of non-convergence. Finally, Figure 4.1 shows a visualization of trajectories generated in four example scenarios. These examples show that both fixed-resolution variants produce poor and unsteady trajectories due to their limited perceptive range. Combining all the presented results, the proposed multi-resolution, purely reactive, hierarchical algorithm provides an attractive balance between success rate, trajectory quality, and computational cost.

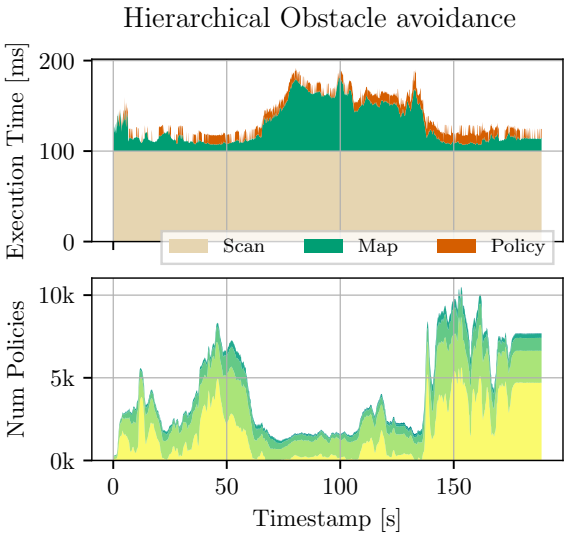


Figure 4.11: Top: Stackplot of data latency (LiDAR) and processing latency (Map/Policy). Bottom: Visualization of the number of policies at different levels, where yellow is the finest resolution and dark green is the coarsest, in similar fashion to Figure 4.3. The system is on the outdoor terrace between $55\,\mathrm{s} - 130\,\mathrm{s}$. Especially after the robot reenters the building, it is in close proximity to many obstacles, leading to more policies at a higher resolution.

4.5.2 Computational efficiency

We benchmark the computational efficiency of the proposed navigation system on a NVidia Jetson Orin AGX computer, using data from a Livox Mid-360 LiDAR. The navigation algorithm only uses the computer's 12-core ARM Cortex-A78AE CPU. Figure 4.11 visualizes the latency and policy processing times on a dataset that traverses indoor offices before transitioning to a terrace, including a large $30\,\mathrm{m}$ radius open space. Together, the mapping and planning use 2.4 CPU threads (average) and $355\,\mathrm{MB}$ of RAM (peak). The LiDAR delivers new data every $100\,\mathrm{ms}$ and integrating these observations takes $29\,\mathrm{ms}$ (average), while selecting and executing the obstacle avoidance policies takes $6.9\,\mathrm{ms}$ (average). All together, the mapping and planning steps are completed almost instantaneously after the LiDAR data is received.

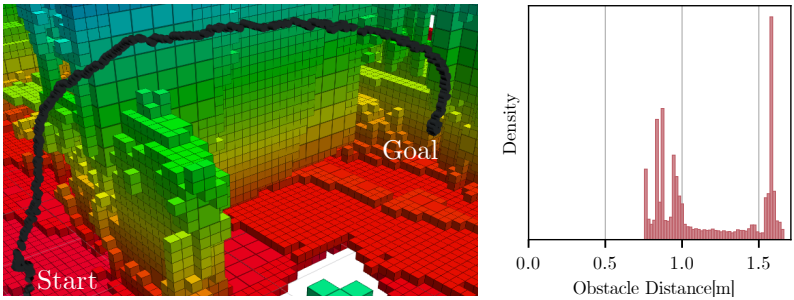


Figure 4.12: Left: Rendering of an executed flight path (black) and the map that was created during a traversal of a cluttered region with the specified goal location. Right: Obstacle distance histogram for the same flight. The MAV successfully cleared all obstacles with sufficient margin. Note: For operational reasons, the tuning for the field test was more conservative (stronger) than for the map-based evaluation.

4.5.3 Platform tests

The proposed navigation pipeline (Figure 4.2) is deployed on an MAV with a Livox Mid-360 LiDAR for odometry [67] and mapping. We run the aerial robot through an indoor obstacle course without a prior map, such that all data used for navigation must be gathered and processed on the fly. The operator sets a desired goal location prior to the flight, which the robot then autonomously tries to reach using the proposed reactive navigation algorithm. Figure 4.12 visualizes a typical path taken by the aerial robot to avoid an obstacle and fly towards a (potentially unreachable) goal position. Upon setting a desired goal position, the goal-seeking policy starts to drive the robot. After about $130\,\mathrm{ms}$, the first scan is received, the map is populated and the obstacle avoidance policies become active. As can be seen from Figure 4.12, the robot avoids the obstacles with sufficient distance. During the full run, the robot never got closer than $0.75\,\mathrm{m}$ to an obstacle and kept an average closest-obstacle distance of $1.16\,\mathrm{m} \pm 0.32\,\mathrm{m}$.

4.6 Conclusion

In this chapter, we presented a novel method for multi-resolution, reactive obstacle avoidance in generic 3D environments. A key insight is the efficient use of multi-resolution, hierarchical obstacle information. This follows the intuition that geometry further away does not need to be incorporated at the same resolution as nearby obstacles. As demonstrated through numerical analysis and ablations, the proposed approach enables locally precise and safe collision avoidance while keeping a very large perceptive radius. Multi-resolution obstacles can efficiently be extracted by directly exploiting the hierarchical structure present in hierarchical volumetric mapping frameworks such as wavemap [7]. The proposed system achieves planning success rates comparable to CHOMP while reducing the planning time by $50\times$ and requiring no pre-processing or post-processing steps, such as EDF generation and trajectory tracking control. Finally, the system is deployed on a real MAV negotiating an indoor obstacle course while only using minimal computational resources.

Chapter 5

Multi-resolution global planning

Being able to find a path that allows a robot to travel from one point to another in a known environment is a core competency of most robotic systems. Typically, this is solved in a two-step process. First, a global path is computed, which in a second step is followed by a local method. The previous chapter showed how multi-resolution maps enable safe and efficient local navigation. This obviously begs the question of whether exploiting such multi-resolution representations for global planning is possible as well. A multi-resolution map captures the connectivity between large regions of space with explicit knowledge about their occupancy status. Commonly used sampling and trajectory optimization methods do not take advantage of this information. Search-based methods such as A* can make use of such connectivity information but do not scale well computationally. Intuitively, paths composed of straight lines link corners of obstacles, which is exploited by any-angle search-based methods. We extend this idea into a multi-resolution paradigm, representing the map as well as the search information using multi-resolution data structures. Experiments demonstrate how the proposed approach achieves results on par with fixed-resolution methods while being several orders of magnitude faster, beating even sampling-based methods. An open-source implementation of the method is available[1].

[1] https://github.com/ethz-asl/wavefinder

5.1 Introduction

A core competency of robots is the ability to autonomously navigate between areas of interest, such as storage spaces, work sites, and inspection points, even if these locations are far apart. Methods that find a globally optimal solution to this planning problem can be categorized into sampling and search-based methods. Both categories extract the shortest path from a graph representing a discretization of the robot's configuration and transition space. While search-based methods generally operate on a graph with a fixed topology, such as a grid map's adjacency graph or a state lattice constructed from motion primitives, sampling-based methods build the graph by randomly sampling and connecting collision-free robot configurations. Sampling-based solutions are popular in practice due to their ability to find solutions while only sparsely covering large, possibly high-dimensional configuration spaces. However, extracting a graph from a volumetric map through random sampling discards a lot of the information embedded in the map and neglects its underlying structure. The amount of data contained in a discretized map is finite, yet sampling-based methods are only asymptotically complete and cannot report infeasibility in finite time. This is particularly problematic in environments with narrow passages, where solving a feasible planning query can take a long time, and feasibility is not guaranteed. This gives rise to a hard-to-answer question in sampling-based methods, namely: How long should one try to find a solution before giving up?

In many applications, including long-range navigation, a volumetric map's adjacency graph provides a reasonable representation of the transition graph. Pairing the adjacency graph with a standard graph search algorithm results in a planner that is both resolution complete and able to report whether a solution exists in finite time. While searching for the shortest path, methods based on Dijkstra's algorithm [68], including A* [69], will explicitly compute the optimal cost-to-come and predecessor for each explored grid vertex. This creates a cost field that can be interpreted as a volumetric property of the environment. Similar to other volumetric methods, including occupancy mapping and ESDF generation, the time and space complexity of running A* on a fixed-resolution grid scales linearly with the number of explored grid cells[2]. Therefore, the cost grows linearly with the explored volume and cubically with the grid resolution, which is problematic when the search has to overcome local minima, such as dead-ends, that are large compared to the chosen resolution. In this chapter, we investigate how a multi-resolution map representation can be used to overcome this limitation.

[2]When using a bucket queue for the min-priority queue.

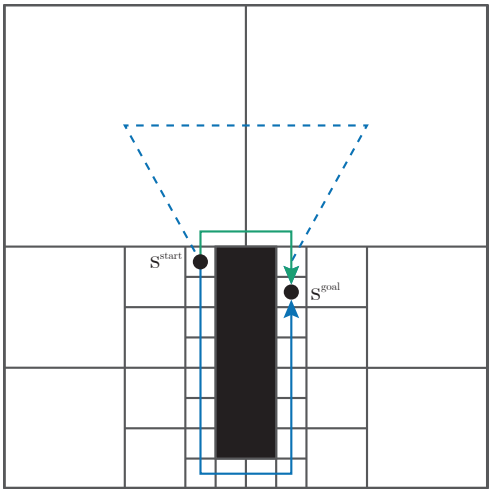


Figure 5.1: Illustration of the sub-optimality of A* when applied to an octree's leaves while only considering their center points, adapted from [70]. The path found by A* (solid blue) is significantly longer than the true shortest path (green). Although the octree completely captures the free space (white) and a path belonging to the right homotopy class is available (dashed blue), it is ignored by A* because the leaves' centers introduce a significant detour.

Octrees are commonly used to compactly encode an environment's occupancy or traversability information using multi-resolution. Early works show that running A* directly on an octree's adjacency graph [61] is resolution complete. However, only considering the centers of the octree's nodes results in paths that are far from optimal in terms of length and smoothness [70]. Since the retrieved shortest path often does not even fall in the same homotopy class as the true shortest path, as illustrated in Figure 5.1, this issue cannot be overcome with traditional post-processing steps such as path-shortening. Several research efforts have proposed to interleave the global search with local approaches to find better paths through the interiors of the octree's free leaf cells. In contrast to such methods, we propose a method that only requires a single, global search routine. We combine a novel multi-resolution cost field representation with an algorithm to dynamically adjust the search resolution where needed, enabling us to accurately plan through large free-space areas.

By construction, adjacent vertices in the cost field rarely have the same cost-to-come (g cost). Using multi-resolution to encode the g costs directly, therefore, does not significantly reduce the number of cells unless high approximation errors are acceptable. When search-based planners are constrained to plan along grid edges, neighboring vertices generally also have different predecessors. However, true shortest paths in continuous space are straight except at points where they tightly wrap obstacles. Based on this insight, any-angle planning algorithms aim to find shorter paths on grid maps by allowing the path to deviate from the grid's edges. One such algorithm, Theta* [6], attempts to set each vertex's predecessor to the best vertex within its line of sight, without constraining it to be a direct neighbor. As shown by Nash et al. [71], this allows Theta* to find paths that are up to $\approx 13\%$ shorter than A* solutions. Another interesting consequence is that large subregions of the cost field tend to point to identical predecessors. Our proposed multi-resolution representation leverages this property to model the cost field accurately and memory-efficiently.

Although improving accuracy and memory efficiency is important, the main bottleneck for search-based planners operating in large environments is often runtime. The key question of this chapter is, therefore, how multi-resolution can be used to speed up the search. Concretely, we want to reduce the number of multi-resolution cells expanded by the planner by exploring each region of the search space at the lowest possible resolution. We present a practical algorithm that achieves this by extending Theta* to operate on an octree structure. The search cost field initially covers the entire search space at the lowest resolution. During the search, the resolution is increased in a coarse to fine manner where needed by dynamically subdividing nodes until each node has a single dominant predecessor. Finally, we present a simple procedure to ensure that all inflection points that could be optimal predecessors are efficiently considered.

In summary, the main contribution of this chapter is a global planner that uses:

- A multi-resolution cost field representation that combines the accuracy of any-angle planning methods with the memory efficiency of octree-based representations;
- A complimentary algorithm that improves computational efficiency by dynamically adjusting the search resolution, in a coarse-to-fine manner, to control the worst-case approximation error;
- An initialization routine that allows our multi-resolution planner to achieve accuracy in line with that of Theta* running at the highest resolution;

The proposed multi-resolution planner is extensively evaluated in a variety of real indoor and outdoor environments. Through ablations, we quantify the properties of

our method's core components and show how they allow users to intuitively control the worst-case sub-optimality of the solution with respect to Theta* running at the highest resolution. We further compare our proposed planner to a range of well-established search and sampling-based planners. The results demonstrate that it reliably finds shorter paths than RRTConnect and RRT* in all environments and achieves significantly higher success rates in confined spaces. Furthermore, the results empirically show that our method maintains the completeness guarantees of high-resolution search-based planners while being significantly faster, both in finding a solution and reporting that none exists. In particular, our method runs 1 to 3 orders of magnitude faster than Theta* when allowed to find solutions that are longer by 2% at worst and 0.1% on average. The entire framework will be open-sourced to allow the robotics and planning communities to build on these results.

5.2 Related work

Path planning methods can generally be categorized into sampling- and search-based approaches. Sampling-based methods are commonly used for global planning, especially in large environments. While very fast, randomized variants such as RRT [72] and RRTConnect [73] are non-deterministic and provide no guarantees on the quality of their solutions. Optimal variants, such as RRT* [74], are guaranteed to converge to the optimal solution as the number of samples grows to infinity. However, they do not provide bounds on the quality of their intermediate solutions. Stopping them after a limited amount of time leads to different solutions, even when the start and goal positions are the same. This inconsistency is due to their stochastic nature and generally worsens as the number of obstacles in the environment increases. A challenge in practice is that randomized planners can take a very long time to find any solution in environments with narrow passages.

Search-based planners, such as A* [69], are directly applied to a specific space discretization. Given their deterministic nature, they are perfectly consistent, and in bounded spaces, they terminate in finite time. Furthermore, they are complete and explicitly report when no solution exists within their discretized space. When an occupancy map is used to represent the environment, a common choice is to apply the search directly to its adjacency graph. Unfortunately, search-based planners suffer from the curse of dimensionality and are expensive to run on large, high-resolution 3D maps.

Several research efforts have investigated the use of hierarchical approaches to improve the scalability of search-based planning. Kambhampati et al. [61] used an octree

to compactly represent the free space and showed how running A* directly over the octree's leaves yields significant efficiency improvements, albeit at the cost of significantly longer, jagged paths. Funk et al.[64] show how this approach can be extended to allow efficient orientation-aware planning through large environments with narrow openings. Instead, CFA* [75] uses only two resolution levels and obtains paths that are comparable to those of A* running at the highest resolution by performing an initial search over coarse blocks, which is then used to restrict a final search at the grid cell level. In a more general vein, HPA* [76] performs a coarse-to-fine search over clusters with pre-computed traversal costs, which can be generalized to an arbitrary number of hierarchical levels but requires more pre-processing. Iterative [77] and information theoretic [78] methods have also been proposed. Recently, Du et al. [62] showed how multiple weighted-A* searches, running simultaneously at different resolution levels, can share information to combine their strengths. An important drawback of all the aforementioned methods is that they are sub-optimal or, at best, equivalent to A* running on the highest resolution grid.

Any-angle planning algorithms are variants of A* that can find solutions that are up to $\approx 13\%$ shorter by allowing the path to deviate from the grid's edges [71]. Intuitively, these deviations allow the path to get closer to the true shortest paths in continuous space, which are taut, i.e., straight except at inflection points where they tightly wrap an obstacle. Just like A*, any-angle planners only propagate information along grid edges which allows for an efficient implementation. Theta* [6] is popular in practice due to its simplicity and ability to reliably find very short paths in diverse environments [79]. However, it performs a large number of visibility checks to verify whether each considered grid deviation is collision-free, which introduces a significant runtime overhead in 3D. LazyTheta* [71] shows how lazy visibility checking can reduce the overhead by an order of magnitude without significantly affecting the resulting paths.

Multi-resolution methods have also been employed for efficient any-angle planning. For example, early work by Chen et al. [70] proposed to plan on framed quadtrees, i.e., quadtrees whose leaf nodes are padded with high-resolution vertices. By allowing the path to take on a broad range of angles through each leaf, this approach effectively finds any-angle paths. Unfortunately, the connectivity within each leaf grows quadratically with the chosen maximum resolution in 2D and quartically in 3D. Although Chen et al. show how this complexity can be overcome in 2D using a linear-time dynamic Voronoi diagram computation algorithm, efficient generalizations to 3D remain an open problem. Closest to our work is the global planner proposed by Faria et al. [80], which uses an octree-based occupancy map to decompose the free space and applies LazyTheta* to its leaves. In a similar fashion, we extend Theta* (and LazyTheta*) to

operate on an octree. However, in contrast to their method, we show that decoupling the planner's resolution from the occupancy map's octree allows it to find better paths. Furthermore, we introduce an initialization procedure that closes the sub-optimality gap with respect to Theta* running at the highest resolution.

5.3 Overview of Theta*

Given that our method's cost field formulation and search algorithm can be seen as a multi-resolution extension of Theta*, we briefly introduce it in this section. Theta* [6] is an any-angle path-finding extension to the A* [69] search algorithm. Just like A*, it only propagates information along grid edges. The key distinction between the two search algorithms lies in how they select each vertex's predecessor. Since A* only considers each vertex's direct neighbors, the paths it returns are strictly composed of grid edges. Theta* additionally considers connections to each direct neighbor's best predecessor, if it is within the vertex's line of sight. This allows Theta* to deviate from the grid and find paths that are up to $\approx 13\%$ shorter than those found by A*, at the cost of increased runtime due to the additional visibility checks.

The main loop of A* and Theta*, shown in Algorithm 3, is identical. Both search algorithms store two values per vertex, namely the vertex's cost-to-come (g cost) and an index or pointer to its best `predecessor`. Furthermore, both algorithms use a priority queue (`open`) to expand vertices in order of their minimum f score, where $f(s) = g(s) + h(s)$ with $h(s)$ a consistent heuristic function. Using a consistent heuristic guarantees that a node is only expanded from the queue once its optimal g cost and `predecessor` have been found [81]. A closed set (`closed`) can therefore be used to track and explicitly skip updates of already expanded nodes (Line 13). It also means that both algorithms can terminate immediately once the goal vertex s^{goal} is expanded (Line 8). Since the paths found by A* can only contain edges of the 26-connected grid, using the octile distance to the goal, $h(s) = \|s^{\text{goal}} - s\|_{\text{oct}}$, is consistent. In contrast, Theta* must use the Euclidean distance, $h(s) = \|s^{\text{goal}} - s\|_2$, because its paths are not constrained to the grid's edges.

As highlighted earlier, the key difference between A* and Theta* is how they find each vertex's best predecessor. When expanding vertex s, function `UpdateVertexCost` is called for each neighboring vertex s' to check if using s could lead to a shorter path. In that check A* only considers connecting s' directly to s (Algorithm 4, note that $c(s^a, s^b)$ refers to the edge cost between two vertices s^a and s^b). As shown in Algorithm 5, Theta* considers connections from s' to both s and `predecessor`(s). By virtue of the triangle inequality, a connection to `predecessor`(s) – when

Algorithm 3: Heuristic-guided search over graph vertices

```
open ← ∅
closed ← ∅
g(s^start) ← 0
predecessor(s^start) ← s^start
open.insert(s^start, g(s^start) + h(s^start))
while open ≠ ∅ do
    s ← open.pop()
    if s = s^goal then
        return PathFound
    end
    closed ← closed ∪ {s}
    foreach s' ∈ neighbors(s) do
        if s' ∉ closed then
            if s' ∉ open then
                g(s') ← ∞
                predecessor(s') ← NULL
            end
            UpdateVertex(s, s')
        end
    end
end
return NoPathFound
Function UpdateVertex(s, s') is
    Status ← UpdateVertexCost(s, s')
    if Status = Changed then
        if s' ∈ open then
            open.remove(s')
        end
        open.insert(s', g(s') + h(s'))
    end
end
```

Algorithm 4: Definitions for A*

```
1  Function UpdateVertexCost(s, s') is
2      if g(s) + c(s, s') < g(s') then
3          predecessor(s') ← s
4          g(s') ← g(s) + c(s, s')
5          return Changed
6      end
7      return Unchanged
8  end
```

Algorithm 5: Definitions for Theta*

```
1  Function UpdateVertexCost(s, s') is
2      s^p ← predecessor(s)
3      if LineOfSight(s^p, s') then
           // Evaluate ray traced connection
4          if g(s^p) + c(s^p, s') < g(s') then
5              predecessor(s') ← s^p
6              g(s') ← g(s^p) + c(s, s')
7              return Changed
8          end
9      else
           // Evaluate direct neighborhood connection
10         if g(s) + c(s, s') < g(s') then
11             predecessor(s') ← s
12             g(s') ← g(s) + c(s, s')
13             return Changed
14         end
15     end
16     return Unchanged
17 end
```

available – is guaranteed to yield a candidate g cost that is equal to or lower than a direct connection to s. Therefore, Theta* only considers connecting to direct neighbor s when its $\texttt{predecessor}(s)$ is not visible from s'.

5.4 Method

In the following, we describe the components of our approach. We start by describing our multi-resolution cost field representation. Next, we present a complementary search algorithm that explores the search space in a coarse to fine manner. Finally, we explain how the paths found by the planner can be improved further without significantly decreasing the planner's efficiency using a dedicated inflection point initialization procedure.

5.4.1 Multi-resolution cost field representation

When applied directly to a grid-based map's adjacency graph, planners such as A* and Theta* compute the cost-to-come and best predecessor for all grid vertices that are explored during the search. As motivated in the introduction, the g cost and $\texttt{predecessor}$ fields can, therefore, be interpreted as volumetric properties. Since both properties are often stored together, going forward, we will simply refer to their combination as 'the cost field'. In this section, we introduce how multi-resolution can be used to encode the cost field more efficiently without sacrificing accuracy.

By construction, neighboring grid vertices rarely share the same g cost. In contrast, the $\texttt{predecessor}$ field of Theta* tends to contain many constant subregions where all grid vertices point to the same dominant inflection point, as illustrated in Figure 5.2. Many subregions of the cost field of Theta* could, therefore, be represented at a lower resolution without affecting the accuracy of the planner. Concretely, we propose to divide the cost field into subvolumes $\mathcal{V}$ and, for each subvolume, only store $\texttt{predecessor}(\mathcal{V})$ and $g(\texttt{predecessor}(\mathcal{V}))$. The g cost of each vertex s in subvolume $\mathcal{V}$ is then stored implicitly, and can be obtained by evaluating

$$\hat{g}(s) = g\big(\texttt{predecessor}(\mathcal{V})\big) + c\big(\texttt{predecessor}(\mathcal{V}), s\big). \tag{5.1}$$

We further propose to organize the subvolumes such that they correspond to an octree's leaf nodes. The main motivation for this is that octrees have a regular structure and that their leaves divide the space into non-overlapping partitions, which simplifies neighborhood operations. Since octree data structures support efficient random access

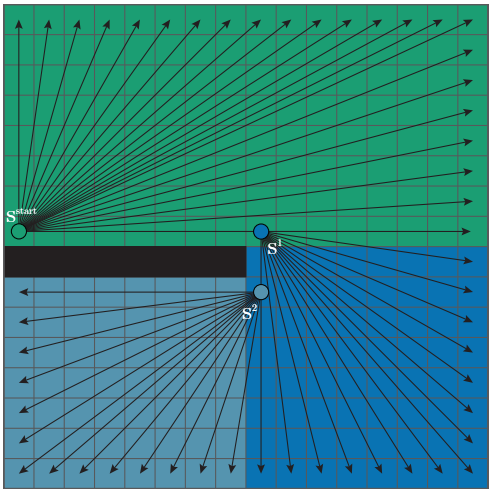


Figure 5.2: Illustration of the `predecessor` field of Theta* for a 2D environment with a single obstacle (black box). All vertices in the upper half of the environment are directly visible from the start vertex (green circle) and thus use s^{start} as their predecessor. Due to the obstacle, paths to the bottom right of the environment are no longer visible from s^{start} and instead use inflection point s^1 (blue circle). Finally, paths to the bottom left of the environment pass through s^2 (light blue circle). Each grid cell is colored according to its predecessor. As can be seen, the field is mostly constant.

and insertions, it is possible to not only store but also generate the cost field more efficiently using multi-resolution. Ideally, we want to explore each region of the search space at the lowest possible resolution to minimize the number of (multi-resolution) cells that have to be processed by the planner. The hierarchical structure of octrees can be exploited to initially represent the entire cost field at the lowest resolution and dynamically increase the resolution only where needed. In other words, they make it tractable to generate the cost field in a coarse to fine manner.

As shown in Figure 5.3, Theta*'s predecessor field is constant in areas that have direct visibility to the start vertex s^{start} but breaks into many thinner constant subregions after wrapping around the first obstacle's edges. These thin, Voronoi-like regions, which are dominated by a single inflection point, can extend forever. However, as the distance of a subvolume $\mathcal{V}$ to two conflicting inflection points that are adjacent

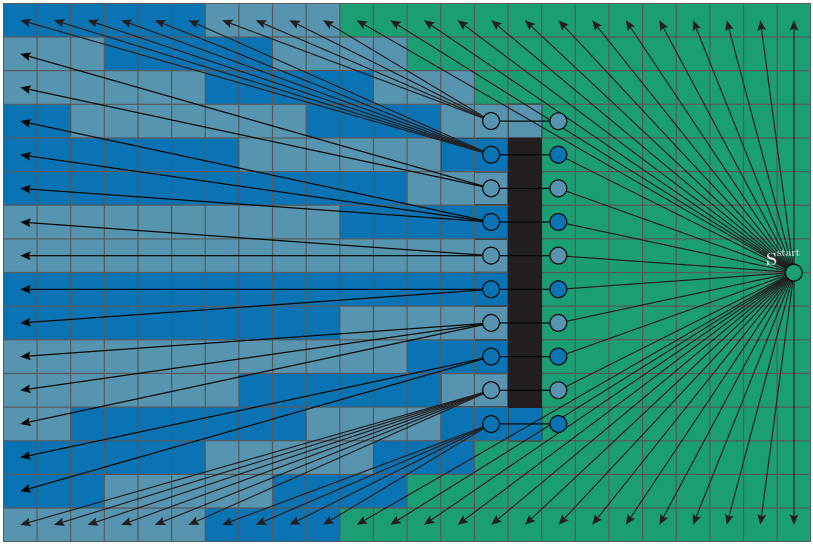


Figure 5.3: Top view showing a 2D slice of the `predecessor` field of Theta* planning through a 3D environment with a single obstacle (black box). As can be seen, the field is initially constant (green region). However, once the paths pass over the obstacle, direct visibility to the start vertex, s^{start}, is lost, and the field splits into many thin, Voronoi-like subregions (alternating dark and light blue), each dominated by a different inflection point (black circle) on the obstacle's border.

to each other increases, the importance of choosing one inflection point over the other quickly decreases. By tolerating very small sub-optimalities, we can therefore represent wide-open free space regions at a significantly lower resolution. Concretely, we propose to quantify the approximation error over a subvolume $\mathcal{V}$ conservatively, as

$$\begin{aligned} E(\mathcal{V}) &= \max_{s \in \mathcal{V}} \frac{\left|\hat{g}(s) - \bar{g}(s)\right|}{\bar{g}(s)} \\ &= \max_{s \in \mathcal{V}} \frac{1}{\bar{g}(s)} \left| g\big(\texttt{predecessor}(\mathcal{V})\big) + c\big(\texttt{predecessor}(\mathcal{V}), s\big) - \bar{g}(s) \right| \end{aligned} \tag{5.2}$$

where $\bar{g}(s)$ refers to the cost that would be obtained by Theta* running at the maximum resolution and $\hat{g}(s)$ refers to the cost extracted from our multi-resolution representation

through Equation (5.1).

Algorithm 6: Heuristic-guided search over octree nodes

1 `open` $\leftarrow \emptyset$
2 `closed` $\leftarrow \emptyset$
3 $g(s^{\text{start}}) \leftarrow 0$
4 `predecessor`$(\mathcal{V}^{\text{start}}) \leftarrow s^{\text{start}}$
5 `open`.$insert(\mathcal{V}^{\text{start}}$, `ComputeFScore`$(\mathcal{V}^{\text{start}}))$
6 **while** *open* $\neq \emptyset$ **do**
7 $\quad \mathcal{V} \leftarrow$ `open`.$pop()$
8 $\quad$ **if** $s^{goal} \in \mathcal{V}$ **then**
9 $\quad\quad$ **return** *PathFound*
10 $\quad$ **end**
11 $\quad$ `closed` $\leftarrow$ `closed` $\cup \{\mathcal{V}\}$
12 $\quad g(\mathcal{V}_{\text{center}}) \leftarrow g($`predecessor`$(\mathcal{V})) + c($`predecessor`$(\mathcal{V}), \mathcal{V}_{\text{center}})$
13 $\quad$ `UpdateNode`$(\mathcal{V}, \mathcal{V}_{\text{root}})$
14 **end**
15 **return** *NoPathFound*

5.4.2 Multi-resolution search

We now present our approach to efficiently generate multi-resolution cost fields in a coarse to fine manner, dynamically increasing the resolution where needed to keep the approximation error below a user-defined threshold ϵ. We start by discussing how heuristic-guided search can be applied to an octree's nodes. Thereafter, we present the specific subroutines that are needed to realize our multi-resolution formulation of Theta*.

The main loop of our multi-resolution planner, shown in Algorithm 6, follows the same general structure as the main loop of A* and Theta* (Algorithm 3). The most obvious difference is that it now operates on subvolumes $\mathcal{V}$ instead of individual vertices s. For each expanded subvolume $\mathcal{V}$, we first check if it contains the goal vertex s^{goal} (Line 8), in which case the search terminates, otherwise, we mark $\mathcal{V}$ as closed. On Line 12, we compute and store the g cost for the vertex at the subvolume's center ($\mathcal{V}_{\text{center}}$). This is necessary because $\mathcal{V}_{\text{center}}$ might itself become a `predecessor` in the future, and we store the g cost of each `predecessor` explicitly. Finally, we process every subvolume $\mathcal{V}'$ that is adjacent to $\mathcal{V}$, to evaluate whether the cost of $\mathcal{V}'$

can be improved. As motivated earlier, the cost field is generated in a coarse to fine manner. This can be implemented using a recursive `UpdateNode` function, which always starts at the octree's root node ($\mathcal{V}_{\text{root}}$).

The `UpdateNode` function is shown in Algorithm 7. It recursively visits every multi-resolution neighbor $\mathcal{V}'$ of the expanded subvolume $\mathcal{V}$ and evaluates whether information from $\mathcal{V}$ could be used to improve the path to $\mathcal{V}'$. It also dynamically increases the resolution of the cost field to keep the worst-case approximation error below a user-defined threshold ϵ. The function's logic consists of two main parts.

The first part (Line 2) is only executed if the current subvolume $\mathcal{V}'$ is a leaf of the octree and calls `UpdateNodeCost`, which tests whether the g cost of some or all of the vertices in the subvolume can be improved based on information from $\mathcal{V}$. Depending on the outcome, `UpdateNodeCost` can update the `predecessor` of $\mathcal{V}'$ and return *Changed*, do nothing and return *Unchanged*, or determine that the best `predecessor` is ambiguous and return *ShouldRefine*. If `UpdateNodeCost` changes the `predecessor`, we update the priority of subvolume $\mathcal{V}'$ in the `open` queue based on its new f score. When `UpdateNodeCost` returns *Unchanged*, no action is required. These first two cases imply that the current resolution is adequate, and no further recursion is required. Finally, if `UpdateNodeCost` returns *ShouldRefine*, we remove $\mathcal{V}'$ from the priority queue at its current resolution, and the control flow proceeds to the for-loop on Line 16.

The second part of `UpdateNode` (Line 16) is executed if subvolume $\mathcal{V}'$ has already been refined in the past, or was marked for refinement by `UpdateNodeCost`, see above. Since we use an octree subdivision scheme, each subvolume $\mathcal{V}'$ has exactly 8 children. Note that updates to children that have already been `closed` are skipped, for the same reasons as in A* and Theta*. Newly created children inherit their parent's predecessor $s^{p'}$ and are directly inserted into the open queue. This ensures that the entire volume covered by parent $\mathcal{V}'$, which was removed from the queue on Line 13, will eventually be expanded. The algorithm then recursively visits each child node that is a direct neighbor of $\mathcal{V}$ (Line 23), possibly updating the child's priority in the open queue (Line 8).

We now present the definitions of `UpdateNodeCost` and `ComputeFScore` that extend Theta* to the multi-resolution case. As shown in Algorithm 8, function `UpdateNodeCost` follows the same overall structure as its fixed resolution counterpart, `UpdateVertexCost` (Algorithm 5). The key difference is that, instead of only considering the effect of new connections on the g cost of a single vertex, we now consider all the vertices in subvolume $\mathcal{V}'$. We refer to the subvolume's current

Algorithm 7: Recursive algorithm to update octree nodes

```
1  Function UpdateNode(𝒱, 𝒱′) is
2      if IsLeaf(𝒱′) then
3          Status ← UpdateNodeCost(𝒱, 𝒱′)
4          if Status = Changed then
5              if 𝒱′ ∈ open then
6                  open.remove(𝒱′)
7              end
8              open.insert(𝒱′, ComputeFScore(𝒱′))
9              return
10         else if Status = Unchanged then
11             return
12         else // Status = ShouldRefine
13             open.remove(𝒱′)
14         end
15     end
16     foreach 𝒱′^child ∈ 𝒱′ do
17         if 𝒱′^child ∉ closed then
18             s^p′ ← predecessor(𝒱′)
19             if 𝒱′^child ∉ open then
20                 predecessor(𝒱′^child) ← s^p′
21                 open.insert(𝒱′^child, ComputeFScore(𝒱′^child))
22             end
23             if AreAdjacent(𝒱, 𝒱′^child) and 𝒱′^child ≠ 𝒱 then
24                 UpdateNode(𝒱, 𝒱′^child)
25             end
26         end
27     end
28 end
```

Algorithm 8: Definitions for Multi-Resolution Theta*

Function `UpdateNodeCost`$(\mathcal{V}, \mathcal{V}')$ **is**
  $s \leftarrow \mathcal{V}_{\text{center}},\ s^p \leftarrow$ `predecessor`$(\mathcal{V}),\ s^{p'} \leftarrow$ `predecessor`$(\mathcal{V}')$
  if `LineOfSight`$(s^p, \mathcal{V}')$ **then**
    `// Evaluate ray traced connection`
    if `IsBetterOrSimilar`$(s^{p'}, s^p, \mathcal{V}')$ **then**
      return *Unchanged*
    else if `IsBetterOrSimilar`$(s^p, s^{p'}, \mathcal{V}')$ **then**
      `predecessor`$(\mathcal{V}') \leftarrow s^p$
      return *Changed*
    end
  else
    `// Evaluate direct neighborhood connection`
    if `IsBetterOrSimilar`$(s^{p'}, s, \mathcal{V}')$ **then**
      return *Unchanged*
    else if `IsBetterOrSimilar`$(s, s^{p'}, \mathcal{V}')$ **then**
      `predecessor`$(\mathcal{V}') \leftarrow \mathcal{V}_{\text{center}}$
      return *Changed*
    end
  end
  return *ShouldRefine*
end

Function `IsBetterOrSimilar`$(s^a, s^b, \mathcal{V}')$ **is**
  if $\forall s \in \mathcal{V}' : g(s^a) + c(s^a, s) < g(s^b) + c(s^b, s) + \epsilon\, c(s^a, s)$ **then**
    return *True*
  else
    return *False*
  end
end

Function `ComputeFScore`$(\mathcal{V})$ **is**
  $s^p \leftarrow$ `predecessor`$(\mathcal{V})$
  return $\min_{s \in \mathcal{V}} [g(s^p) + c(s^p, s) + h(s)]$
end

predecessor as $s^{p'}$ and to the candidate predecessors as s^p and s, which correspond to the predecessor and center point of $\mathcal{V}$, respectively. Note that the reasoning behind using $s \leftarrow \mathcal{V}_{\text{center}}$ will be explained in the next section. If keeping the subvolume's current predecessor $s^{p'}$ yields a lower or similar g cost for all the vertices $s \in \mathcal{V}'$ when compared to switching to s^p or s, no changes are required. When the inverse is true for all $s \in \mathcal{V}$, we update the predecessor and return *Changed* to signal that the priority of $\mathcal{V}'$ in the open queue should be updated. In ambiguous cases, where the current predecessor outperforms the candidate by more than ϵ for at least one vertex while the opposite is also true, `UpdateNodeCost` returns *ShouldRefine* to signal that the resolution should be increased to control the approximation error.

The function `IsBetterOrSimilar` (Line 20) formally describes how the predecessors are compared. When ϵ is set to zero, `IsBetterOrSimilar` returns `True` if connecting to vertex s^a is strictly better than connecting to vertex s^b for every single vertex in subvolume $\mathcal{V}'$. Setting $\epsilon = 0$ thus results in our algorithm refining every subvolume until it is strictly dominated by a single vertex. As motivated in the previous sections, our planner's efficiency can be improved by tolerating small sub-optimalities as quantified by Equation (5.2). However, to interleave the refinement with the search itself, the sub-optimality must be estimated incrementally. We propose to estimate the worst-case suboptimality of predecessor $s^{p'}$ with respect to s^p using

$$\hat{E}\big(s^{p'}, s^p, \mathcal{V}'\big) = \max_{s \in \mathcal{V}} \frac{g(s^{p'}) + c(s^{p'}, s) - g(s^p) - c(s^p, s)}{c(s^{p'}, s)} \tag{5.3}$$

where $c(s^p, s)$ is the straight line distance from s^p to s. Note that this equation quantifies the error for each candidate edge relative to the edge's length, instead of the accumulated g cost as in Equation (5.2). Formally, `UpdateNode` will then recurse until the following invariant is satisfied by every $\mathcal{V}'$ adjacent to $\mathcal{V}$ and $s^a \in \{s^p, s\}$:

$$\hat{E}\big(\texttt{predecessor}\big(\mathcal{V}'\big), s^a, \mathcal{V}'\big) \leq \epsilon \tag{5.4}$$

The intended outcome of bounding the worst-case, relative sub-optimality of every path segment is that it also bounds the worst-case sub-optimality of the path as a whole. Although the experiments we conducted thus far seem to support this hypothesis, we are still investigating whether more formal guarantees can be provided.

Finally, `ComputeFScore` illustrates how our multi-resolution planner generalizes the f score computation to operate on subvolumes instead of single vertices. Note that although `IsBetterOrSimilar` and `ComputeFScore` formally consider all the vertices in $\mathcal{V}'$ or $\mathcal{V}$, only a few critical vertices need to be checked in practice given that $c(s^p, s)$ and $h(s)$ are Euclidean distances.

5.4.3 Initializing inflection points

Another way to interpret what our refinement strategy does is that it detects when a subvolume in the cost field has conflicting predecessors and increases the subvolume's resolution until the conflict is resolved. As shown in the previous section, this refinement strategy can efficiently be integrated into a standard search algorithm. However, a restriction of this approach is that a given vertex will only be considered as a potential predecessor for other nodes after it has been expanded. Furthermore, expanded nodes are added to the `closed` set and are no longer updated. In other words, the refinement strategy only operates based on hindsight. In a fixed resolution setting, this is not a problem since the predecessors, or inflection points, directly correspond to the vertices. In our multi-resolution formulation, however, each subvolume that is not at the highest resolution contains multiple vertices and could, therefore, contain multiple inflection points. This is a situation we would like to avoid, as propagating and considering multiple predecessors per subvolume in `UpdateNode` would be complex and computationally expensive. An alternative would be to detect when a subvolume is first used as a predecessor and subdivide it then. But since it would already have been expanded, this would lead to repeated work. A natural question at this point is what would happen if we only considered a single inflection point for each node, such as its center point. The center point is guaranteed to be sub-optimal unless the node is at the highest resolution, in which case the center point is the only vertex it contains. This is because a path can only be the shortest path if it is taut, i.e. if all of its inflection points are directly adjacent to an obstacle. Fortunately, we can leverage this property to fill in our multi-resolution planner's missing foresight by initializing all the inflection points that could appear on the shortest path ahead of time. This way, we effectively trade some computational cost for path optimality.

Implementing our proposed initialization procedure is straightforward. Every subvolume that is occupied in the occupancy map is padded by high-resolution, allocated nodes in the cost field. Note that since `UpdateNode` (Algorithm 7) only operates on the cost field's leaves (Line 2), simply allocating the cost field nodes containing the candidate inflection points is sufficient to guarantee that they will be updated and eventually expanded at the highest resolution. Note that the initialization procedure can be performed at startup for the whole map or interleaved with the search, for example, by dividing the map into sectors and initializing each sector when it is first touched by `UpdateNode`. In our implementation, we apply the latter approach such that the initialization cost scales with the volume that is explored during the search instead of incurring a constant cost that is proportional to the map's entire volume.

5.5 Experiments

We evaluate our multi-resolution planner on four real maps representing a variety of indoor and outdoor environments. In particular, we use the `Mine`, `Cloister`, `Math`, and `Park` sequences of the Newer College Dataset [49]. The Newer College `Mine` sequence was chosen to represent a constrained indoor environment consisting of rooms connected by narrow passages. The `Cloister` sequence features an indoor space with arches, structured obstacles, and tight doorways connected to two wide-open outdoor spaces. `Math` represents a large, structured outdoor urban environment, while `Park` represents an even larger unstructured outdoor environment.

A multi-resolution occupancy map of each environment was generated by *wavemap*, running at a maximum resolution of 10cm and using odometry estimated by FastLIO2 [50]. The obstacles in the map are inflated by 35cm to account for the robot's radius. For each map, we randomly sample 100 pairs of collision-free start and goal positions (400 in total). Note that start-goal position pairs corresponding to infeasible planning queries are not filtered out, since we are also interested in seeing how efficiently the planner can report whether or not a solution exists.

In the following sections, we first present ablations quantifying the behavior of the approximation-error-driven refinement strategy presented in Section 5.4.2, and the initialization strategy presented in Section 5.4.3. We then compare the success rates, path quality, and runtime of our proposed multi-resolution planner to a representative selection of search and sampling-based planners.

5.5.1 Ablations

Refinement strategy

To evaluate the effect of our proposed refinement strategy, we run our planner with different approximation error thresholds ϵ (Eq. (5.4)). We then compare the resulting path lengths and execution times to Theta* running at the highest resolution. Two special cases are included. In the first case, we set $\epsilon = 0$, which forces the planner to subdivide each node in the cost field until it is strictly dominated by a single predecessor. The second case represents the other extreme where refinement is disabled altogether. Note that we use an octree-based occupancy map to represent the traversable space and the planner is only allowed to plan through fully traversable nodes. In a compressed octree, fully traversable nodes will always be leaves. At each point in space, the resolution of the cost field will therefore be at least as high as the

map's leaf resolution. When refinement is disabled the resolutions are equal, which is why we refer to the second special case as `Match map`.

In general, the results in Figure 5.4 show that as threshold ϵ is decreased, the planner finds shorter paths at the cost of increased runtime. It also shows that the path lengths are very close to those of `Theta*` on average, especially for longer paths (deep purple). Most of the outliers correspond to very short paths (light orange). This is partially explained by the fact that noise gets amplified when paths and runtimes for `Theta*` are very short.

Another important observation is that once ϵ reaches 10^{-1}, the path lengths become comparable to refinement being disabled completely (`Match map`), aside from having fewer outliers. This can be explained by the geometry of real environments and the properties of the sensors used to create the occupancy maps, which makes it unlikely for an occupancy map's leaves to grow very large. Worst-case approximation errors beyond $\epsilon = 10^{-1}$ are therefore rarely encountered in practice, even without refinement. In general, the worst-case approximation error naturally tends to stay well below ϵ for most path segments and even further below ϵ when computed over the entire path. One specific thing to note is that we store the occupancy map and cost field using an optimized octree data structure which limits the maximum size of its nodes to $6.4 \times 6.4 \times 6.4\text{m}$. This has the side effect of avoiding outliers from extremely sub-optimal segments for `Match map`.

Looking at the execution timings in Figure 5.4, we see that the planner is the fastest when refinement is disabled (`Match map`). However, the runtime does not significantly increase when enabling refinement with a relaxed threshold ($\epsilon = 10^{-1}$). We thus recommend always enabling refinement to reduce outliers. As ϵ is reduced further, the runtimes increase but even at $\epsilon = 0$ the planner remains about 1 order of magnitude faster than `Theta*`. This shows that meaningful speedups can be achieved even when the refinement strategy is not allowed to introduce any approximations.

Inflection point initialization

We now quantify the importance of the inflection point initialization procedure presented in Section 5.4.3 by running our planner with and without it. We then compare the resulting path lengths and execution times to `Theta*`. Note that initializing the inflection points at the highest resolution might not be necessary. We, therefore, repeat the experiment for different initialization resolution levels going from 10cm to 6.4m. To isolate the effect of the initialization procedure, we set $\epsilon = 0$ such that the refinement strategy is not allowed to introduce its own approximation errors. The

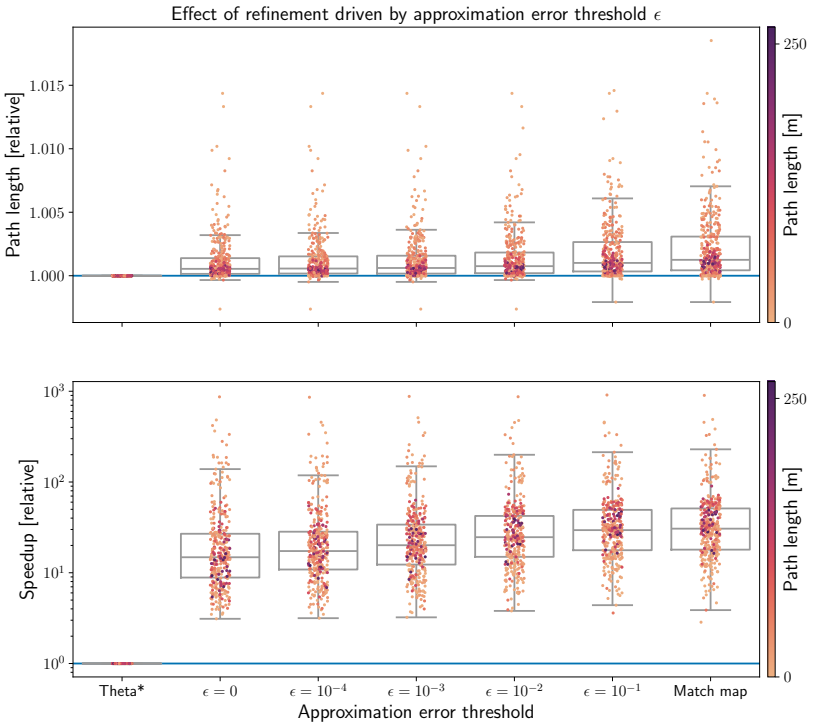


Figure 5.4: Ablation showing how our proposed refinement strategy affects the path length and speedup (log scale) of our method relative to Theta* (blue line). All successful queries on all maps are shown, and each data point is colored by its absolute path length. Reading the plot from right to left, we see that as the threshold is tightened, the path lengths decrease while runtime moderately increases. Note that Theta* is not guaranteed to be optimal, which explains why our method occasionally discovers slightly shorter paths. Finally, notice that even at $\epsilon = 0$, the paths do not match those of Theta*. This is because this ablation does not yet include our inflection point initialization procedure, which is addressed in Figure 5.5.

planner runs without initialization, labeled `No explicit init` in these evaluations, are therefore identical to the runs labeled $\epsilon = 0$ in Figure 5.4.

As shown in Figure 5.5, increasing the inflection point initialization resolution makes the path lengths of our planner converge to those of `Theta*` and significantly reduces the number of outliers. A further observation is that initializing inflection points at low resolutions (1.6m and higher) yields no measurable improvement over `No explicit init`. As explained in the previous subsection, our planner always increases the resolution of the cost field until it matches or surpasses the compressed occupancy map. Furthermore, occupied nodes tend to naturally be surrounded by medium to high-resolution free space nodes. Manually initializing cost field nodes at very low resolutions therefore has no effect.

Looking at the bottom plot of Figure 5.5, we see that increasing the initialization resolution negatively affects runtime. This is not surprising, as it forces the search to expand more cost field nodes and to consider more inflection points as predecessors. One interesting observation is that setting $\epsilon = 0$ and initializing inflection points at the highest resolution allows our planner to find paths that are almost indistinguishable from `Theta*`, while still being significantly faster – by more than one order of magnitude for longer paths (deep purple).

5.5.2 Comparisons

In this section, we compare the success rates, path lengths, and execution times of our proposed multi-resolution planner to a representative sample of search and sampling-based planners. In terms of search-based planners, we implemented fixed-resolution versions of A* [69], `Theta*` [6], and `LazyTheta*` [71]. Note that although the Euclidean and octile distance heuristics are both consistent when A* is applied to a 26-connected grid, the resulting performance varies significantly. We therefore include both `A* Euclidean` and `A* Octile` in our evaluations. For sampling-based planning, we used the RRTConnect [73] and RRT* [74] implementations included in the Open Motion Planning Library [82]. While `RRTConnect` terminates immediately once a path is found, RRT* does not. Therefore, we include three RRT* variants with increasing time budgets, namely `RRT* 0.1s`, `RRT* 1s` and `RRT* 10s`. Note that `RRTConnect` is also limited to a maximum time budget of 10s, to keep it from running forever when a planning query is infeasible.

For our proposed multi-resolution planner, we include three variants: *Ours*, *OursLazy* and *OursFast*. The first variant, *Ours*, exactly matches the algorithm described in the method. The other two variants improve runtime using lazy visibility checking,

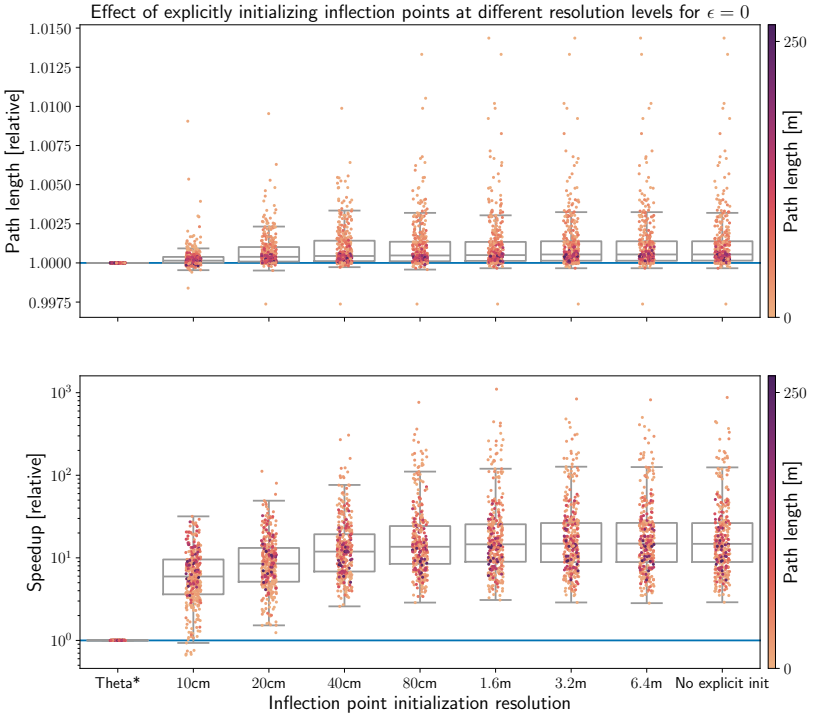


Figure 5.5: Ablation showing how our proposed inflection point initialization procedure affects the path length and speedup (log scale) of our method relative to Theta* (blue line). All successful queries on all maps are shown, and each data point is colored according to its absolute path length. Reading the plot from right to left, we see how explicitly initializing inflection points at a low resolution is equivalent to performing no initialization (`No explicit init`). However, once it exceeds 80cm, increasing the resolution moderately increases runtime and closes the path length sub-optimality gap relative to Theta*. Note how initializing inflection points at the maximum resolution (10cm) allows our method to find paths that are indistinguishable from Theta* while remaining almost one order of magnitude faster.

following the idea of LazyTheta* [71]. Note that this requires minimal changes in the code. The key difference is that on Line 3 of `UpdateNodeCost` (Algorithm 8), we assume `LineOfSight`$(s^p, \mathcal{V}')$ is always true. The search algorithm then checks if this assumption was valid once $\mathcal{V}'$ is expanded (Line 7, Algorithm 6). If it turns out the assumption was false and the visibility check fails, the `predecessor` of $\mathcal{V}'$ is updated to point to the best direct neighbor of $\mathcal{V}'$. The rest of the algorithm remains unchanged. The specific settings we use for our three planner variants are:

- *Ours*: $\epsilon = 10^{-2},\ r^{\text{init}} = 10\text{cm},$ as motivated by the ablations
- *OursLazy*: $\epsilon = 10^{-2},\ r^{\text{init}} = 10\text{cm},$ lazy collision checking
- *OursFast*: $\epsilon = 10^{-2},\ r^{\text{init}} = 40\text{cm},$ lazy collision checking

To make the comparisons as fair as possible, we use the same optimized data structures and subroutines for all planners including ours. The fixed-resolution search-based planners store their cost field using a hashed voxel block data structure, while our multi-resolution planner uses a hashed octree data structure similar to OpenVDB [47]. Every planner uses *wavemap*'s hierarchical occupancy map for fast traversability checking and its multi-resolution ray tracer for fast visibility checking. Finally, all the planners are single-threaded and all the experiments are performed on the same benchmarking server featuring an Intel i9-9900K CPU pinned to 3.6GHz, and 64GB of RAM.

Success rates

Starting with the success rates, shown in Table 5.1, we see that all the search-based planners are equally successful. Note that the evaluated start and goal pose pairs are chosen randomly without filtering out infeasible planning queries. Even complete planners might therefore not always succeed, as is most evident from the `Mine` where none of the planners succeed on more than 89 out of 100 queries. Turning to the sampling-based planners, the highest success rates are achieved by `RRTConnect`, which is almost as successful as the search-based planners. `RRT* 10s` comes in at a close second. Both planners are given a maximum time budget of 10s, but `RRTConnect` probably has a slight edge as it grows trees in both directions and does not spend any time on rewiring. `RRT* 1s` trails shortly behind `RRT* 10s` in easy environments, but its success rate drops significantly once the environment becomes too large (`Park`) or features very narrow passages (`Cloister`). Finally, `RRT* 0.1s` has the lowest success rate of all planners and only performs reasonably well in the `Math` environment, which is large but mainly consists of free space with

Table 5.1: Planning success rates compared across different maps, for 100 randomly sampled start and goal position pairs per map. Note that the queries are not guaranteed to be feasible, which is why even complete planners are not guaranteed to reach 100% success rates.

Success rate (%)	Mine	Cloister	Math	Park
A* Euclidean	89.0	100.0	99.0	99.0
A* Octile	89.0	100.0	99.0	99.0
Theta*	89.0	100.0	99.0	99.0
LazyTheta*	89.0	100.0	99.0	99.0
RRTConnect	89.0	98.0	99.0	99.0
RRT* 0.1s	77.0	46.0	94.0	51.0
RRT* 1s	88.0	55.0	97.0	90.0
RRT* 10s	89.0	84.0	98.0	98.0
Ours	89.0	100.0	99.0	99.0
Ours Lazy	89.0	100.0	99.0	99.0
Ours Fast	89.0	100.0	99.0	99.0

good visibility.

We also verified that for every query for which at least one planner succeeded, every search-based planner succeeded. This empirically suggests that our multi-resolution planners maintain the completeness guarantee of their fixed-resolution counterparts. Furthermore, there were no queries in which a sampling-based planner found a solution while the search-based planners did not. This supports the idea that an occupancy map's adjacency graph provides a reasonable representation of the transition graph for long-range navigation, as suggested in this chapter's introduction.

Path quality

Moving on to the path quality evaluations, we start by comparing the average path lengths for all planners. Note that we only include queries on which all planners succeeded in the averages, to avoid giving an unfair advantage to planners that failed more often on longer paths. As shown in Table 5.2, `Theta*` finds the shortest paths on average on all maps. `LazyTheta*` comes in at a close 2$^{\text{nd}}$ place with paths that are only 0.1% longer on average, and *Ours* achieves comparable results. *OursLazy* and *OursFast* still perform well, with *OursFast* producing slightly longer paths but never falling behind `Theta*` by more than 0.6%. Note that our planners outperform all the

Table 5.2: Path lengths compared across different maps for 100 randomly sampled start and goal position pairs per map. Averaged across all queries for which all planners succeeded.

Mean path length (m)	Mine	Cloister	Math	Park
A* Euclidean	18.21	21.93	43.36	81.9
A* Octile	18.21	21.93	43.36	81.9
Theta*	**16.93**	**20.2**	**40.48**	**76.86**
LazyTheta*	16.95	**20.2**	40.49	76.88
RRTConnect	34.53	44.38	70.58	125.03
RRT* 0.1s	19.99	24.44	42.07	82.14
RRT* 1s	18.0	20.64	41.14	79.26
RRT* 10s	17.29	20.34	40.78	77.57
Ours	16.95	**20.2**	40.49	76.89
Ours Lazy	16.99	20.23	40.53	76.96
Ours Fast	17.04	20.25	40.57	76.99

RRT* variants, whose path lengths gradually increase as their time budgets decrease from `RRT* 10s` to `RRT* 0.1s`. Interestingly, the path lengths of `A* Euclidean` and `A* Octile` lie in between those of `RRT* 0.1s` and `RRT* 1s`. This indicates that, on queries where they succeed, optimizing sampling-based planners such as RRT* can already outperform A* within a fairly short time budget. We interpret this to mean that A*'s worst-case sub-optimality of $\approx 13\%$, which results from only considering the occupancy grid map's 26-connected edges, is quite large in practice. Finally, `RRTConnect` finds the longest paths, which on average are almost twice as long as those of `Theta*`.

Looking at the absolute and relative path length distributions in Figure 5.6, we see that while the random planning queries feature a good spread in terms of evaluated path lengths, ranging from 0 up to 500m, shorter paths are more frequent. This follows from the fact that we sampled 100 trajectories per map and that the `Mine` and `Cloister` maps feature smaller connected areas. In terms of absolute path lengths, most planners find reasonable paths. `RRTConnect` stands out with significantly longer paths, while `RRT* 0.1s` appears to have slightly shorter paths on average. This bias is explained by the fact that `RRT* 0.1s` rarely succeeds when the start and goal are far apart and failed runs are not shown in this plot. Turning our attention to the relative path lengths, we see that `Theta*` very consistently finds the shortest path and is closely followed by `LazyTheta*`. The only planners that occasionally

surpass Theta* are the RRT* variants. The sampling-based planners are, however, the least consistent in terms of optimality. Although the spread of RRT*'s path lengths decreases as the time budget increases, RRT* 10s still features occasional outliers where its solution is 1.8 times longer than Theta*. We also see that the paths of RRTConnect are not only twice as long as those of Theta* on average, but also feature an extremely high variance with worst-case outliers where the path is up to 16.8 times longer. The discrepancy between RRT* and RRTConnect highlights the importance of RRT* improving its paths by rewiring its tree. As predicted by Nash et al. [71], the paths found by A* Euclidean and A* Octile are up to $\approx 13\%$ longer than the true shortest paths. What is interesting is that finding such worst-case paths in practice is not uncommon and that most paths found by A* are at least 2% sub-optimal. Finally, we see that *Ours* is not only close to Theta* on average but also features very few outliers. Running our planner with lazy visibility checking (*OursLazy*) reduces the consistency of its results slightly, and decreasing the inflection point initialization resolution reduces it further. Yet, even *OursFast* still produces paths that would be deemed satisfactory for many applications.

The last path quality metric that we will evaluate is path smoothness. We base our analysis on two metrics that we average across all planned paths: the total curvature and the number of turning points. Both metrics are computed by measuring the angle between the incoming and outgoing edge at each waypoint along the path. To obtain the total curvature, the angles are summed, while the number of turning points simply corresponds to the number of non-zero angles. Although the total curvature mathematically quantifies how the angle changes along the path, the perceived smoothness of a path is highly task and robot-dependent. The number of turning points is therefore included as an alternative, broader metric. Note that many local planners or low-level controllers benefit from having to consider fewer turning points or waypoints, as it gives them more freedom to smooth the path.

The total curvature for each planner and environment is shown in Table 5.3. The most apparent trend is that the paths computed by both A* variants are the roughest by a large margin. This is to be expected because only considering grid edges, that are short and limited to 26 possible angles, results in jagged paths. Note that although A* Euclidean and A* Octile always find paths that are identical in terms of length, A* Octile finds slightly smoother paths as corroborated by other studies [79]. Among all planners, the paths found by Theta* consistently have the lowest curvature. Using multi-resolution planning results in slightly curvier paths and enabling lazy visibility checking or reducing the inflection point initialization resolution increases the curvature further. *Ours* therefore tails Theta* but still outperforms LazyTheta*, which in turn outperforms *OursLazy* and finally *OursFast*. Looking

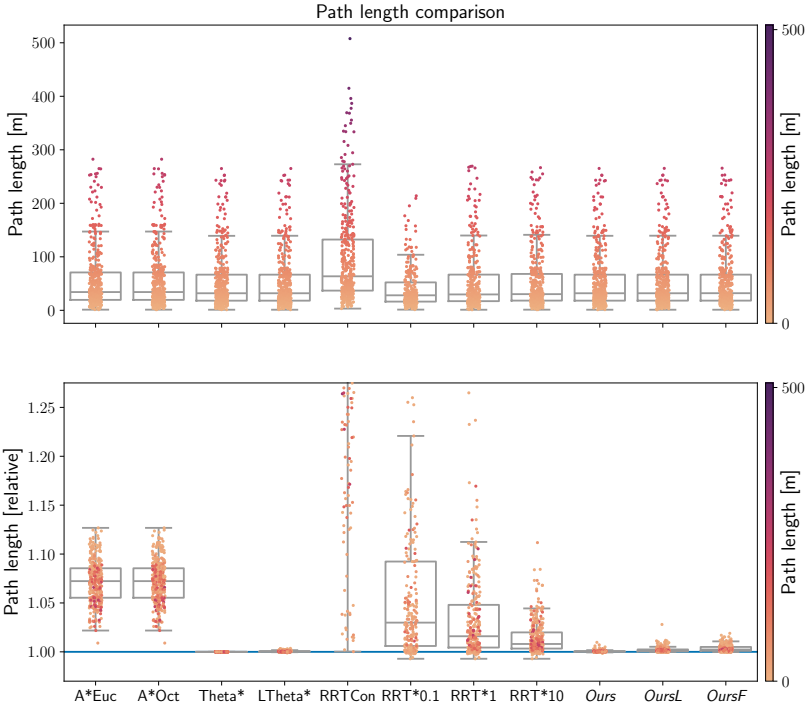


Figure 5.6: Comparison showing the lengths of the paths found by three variants of our multi-resolution planner and a representative selection of search- and sampling-based planners. The upper plot shows the absolute path lengths, while the lower plot shows the path lengths relative to `Theta*` (blue line). All successful queries on all maps are shown. Each data point is colored by its absolute path length, highlighting how most of the outliers in terms of relative path length correspond to very short paths. Note that not all outliers of the sampling-based planners are shown in the lower plot. The paths of `RRTConnect`, in particular, are so long that only the bottom few quantiles are visible.

Table 5.3: Total path curvature across different maps for 100 randomly sampled start and goal position pairs per map. Averaged across all queries for which all planners succeeded.

Mean total curvature (rad)	Mine	Cloister	Math	Park
A* Euclidean	27.52	33.04	50.80	76.30
A* Octile	26.37	29.05	50.09	72.06
Theta*	**1.76**	**0.49**	**1.16**	1.46
LazyTheta*	2.18	0.64	1.34	1.78
RRTConnect	8.81	4.74	5.02	9.23
RRT* 0.1s	2.20	0.79	1.35	1.85
RRT* 1s	1.95	0.61	1.32	**1.39**
RRT* 10s	1.97	0.53	1.45	1.54
Ours	1.97	0.55	1.32	1.69
Ours Lazy	2.57	0.96	1.64	2.38
Ours Fast	2.77	0.90	1.68	2.50

at the sampling-based planners, we see that the smoothest paths are obtained by `RRT* 1s`. We believe this happens since `RRT* 1s` has enough time to find reasonable paths, but does not yet significantly optimize their lengths. Minimum-length paths must tightly wrap the obstacles along their way, which intuitively makes them more sensitive to the geometry of the obstacles' boundaries. Finally, the paths found by `RRTConnect` are significantly rougher than those of RRT*. This is not surprising given that the excess length of `RRTConnect`'s paths has to be folded into the same amount of free space.

Table 5.4 lists the average number of turning points (black) and waypoints (grey) generated by each planner. Generally, the paths found by both A* variants have the highest number of turning points, while those of `RRT* 0.1s` have the fewest. Increasing RRT*'s time budget from `RRT* 0.1s` to `RRT* 10s` allows it to find paths that are shorter, as we saw earlier, but also contain roughly twice as many turns as its paths get closer to the obstacles. The remaining planners, including ours, exhibit this same sensitivity and produce paths whose average number of turning points is approximately equal to `RRT* 10s`.

Table 5.4: Number of turning points (black) and waypoints (grey) across different maps for 100 randomly sampled start and goal position pairs per map. Averaged across all queries for which all planners succeeded.

Mean # of turning points	Mine		Cloister		Math		Park	
A* Euclidean	67	152	85	181	137	371	225	724
A* Octile	66	152	82	181	136	371	222	724
Theta*	6	9	2	4	5	8	8	10
LazyTheta*	7	10	2	4	5	8	8	13
RRTConnect	6	8	3	5	4	6	7	9
RRT* 0.1s	**3**	5	**1**	3	**2**	4	**3**	5
RRT* 1s	4	6	**1**	3	3	5	4	6
RRT* 10s	6	8	2	5	7	9	7	9
Ours	7	10	2	4	6	8	9	11
Ours Lazy	8	11	3	5	6	8	9	12
Ours Fast	7	10	2	4	5	7	8	11

Runtime

The last metric we evaluate is execution time, starting with the average execution time of each planner operating in each environment shown in Table 5.5. Note that all runs are included in the averages, such that unsuccessful queries are also represented, and that the planning times of the RRT* variants are not listed since they are constant. `Theta*` is the slowest of all planners by a large margin. Enabling lazy visibility checking (`LazyTheta*`) improves runtime by 6 to 9 times. However, `LazyTheta*` is still significantly slower than either A* variant. The results also clearly indicate that choosing a heuristic that better confines the explored volume significantly improves runtime, with `A* Octile` being up to 70% faster than `A* Euclidean`. Turning to our proposed planners, we see that leveraging multi-resolution results in a significant speedup. On average, *Ours* is 6 times faster than its fixed resolution counterpart, `Theta*`, in confined environments, and up to 17 times faster in large open spaces. The speedup of *OursLazy* with respect to `LazyTheta*` is slightly smaller but still meaningful, being 2 to 6 times faster. Finally, *OursFast* and `RRTConnect` are tied for first place. While *OursFast* is up to 12 times faster than `RRTConnect` in confined environments such as the `Mine`, `RRTConnect` is up to 14 times faster in large open spaces such as `Park`. Yet, even in the `Park` environment, *OursFast* still performs favorably and is at least 3 times faster than any of the other baseline planners.

We conclude our evaluations by briefly analyzing how the runtimes are distributed.

Table 5.5: Execution times compared across different maps for 100 randomly sampled start and goal position pairs per map. Averaged across all queries. Note that the RRT* variants are omitted as their execution times were fixed to 0.1s, 1.0s, and 10.0s.

Mean execution time (s)	Mine	Cloister	Math	Park
A* Euclidean	0.32	1.58	3.29	12.47
A* Octile	0.24	1.06	2.02	7.41
Theta*	3.16	25.26	65.78	249.22
LazyTheta*	0.5	3.09	7.37	28.51
RRTConnect	1.12	1.36	**0.12**	**0.16**
Ours	0.53	2.15	3.89	18.11
Ours Lazy	0.21	0.69	1.15	5.45
Ours Fast	**0.09**	**0.28**	0.47	2.28

The upper plot in Figure 5.7 shows the absolute execution times for each planner on a logarithmic scale, while the bottom plot shows their execution times relative to Theta* on a linear scale. Following the style of the previous plots, the data points are colored by path length. Looking at the upper plot, we see that for all search-based planners, the execution time is directly correlated to the path length. In contrast, the execution times of `RRTConnect` do not show a meaningful correlation and the execution times of the RRT* variants are constant. In the bottom plot, we see that the runtimes of the search-based planners are strongly correlated to those of `Theta⋆`, while the relative runtimes of the sampling-based planners have much higher variance. The results also show that the speedup of all planners over `Theta⋆` generally increases as the path length increases, emphasizing that `Theta⋆` scales relatively poorly. Looking at our proposed planners, we see that *Ours* and *OursLazy* are rarely slower than `Theta⋆` and can be up to 40 and 120 times faster, respectively. Finally, *OursFast* is strictly faster, yielding speedups of 5 to 1000×.

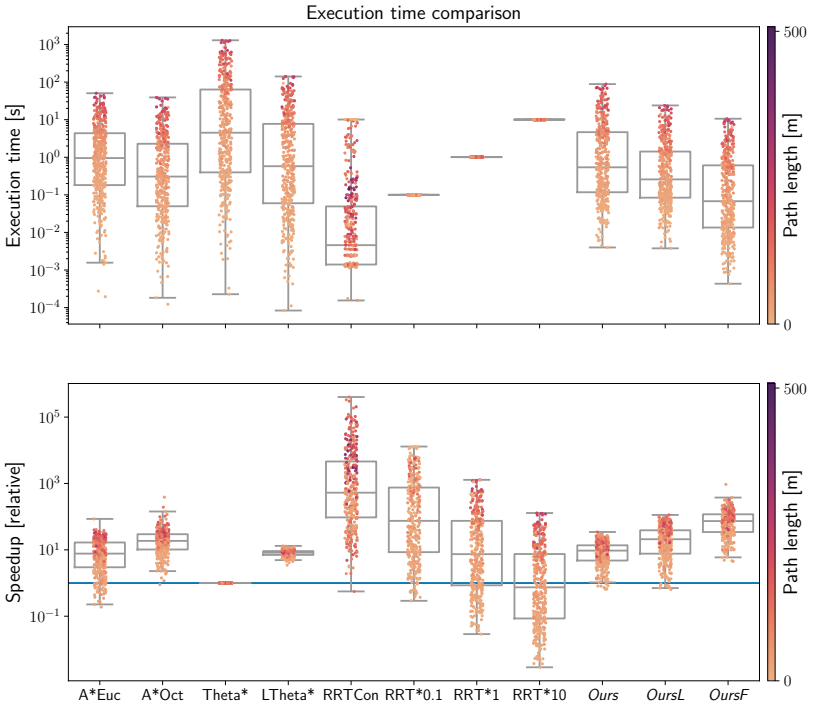


Figure 5.7: Comparison showing the execution times of three variants of our multi-resolution planner and a representative selection of search- and sampling-based planners. The upper plot shows the absolute execution times (log scale), while the lower plot shows the speedups relative to `Theta*` (blue line). All successful queries on all maps are shown. Each data point is colored by its absolute path length, highlighting how the speedup of all planners over `Theta*` grows as the path length increases.

5.6 Conclusion

In this chapter, we presented a search-based global planning method that exploits an octree-like data structure to improve planning speed in occupancy-map environments. We extend the ideas of any-angle planners like Theta* to a hierarchical representation to exploit the sparsity of space. Our proposed method generalizes the concept of inflection points from a fixed-resolution grid representation to a hierarchical one.

Extensive evaluations and comparisons to search-based methods show that we achieve paths of competitive quality but at a substantially reduced computational cost. Showcasing that exploiting the inherent sparsity of real environments does not significantly impact accuracy but provides significant computational benefits. Comparisons to search-based methods demonstrate that our method is capable of detecting feasibility while finding high-quality paths at comparable run times. Together, these results show that our approach, exploiting the spatial structure encoded in a hierarchical map, gives us the benefits and guidance of search-based methods with the speed of sampling-based methods.

Chapter 6

Conclusion and Outlook

This final chapter summarizes the presented work and reflects on the research objectives of this thesis, before discussing interesting avenues for future research.

6.1 Summary

The first research chapter of this thesis introduced an efficient and accurate hierarchical volumetric mapping framework. Using Haar wavelets, it encodes the occupancy posterior as a sparse set of non-zero coefficients that can conveniently be stored using a memory-efficient octree data structure. Beyond memory efficiency, wavelet decompositions guarantee that all resolution levels implicitly remain synchronized. As guaranteed by the MRA conditions, they also ensure that new measurements can safely be integrated into the map in a coarse-to-fine manner. We introduce a hierarchical measurement integrator that uses this fact to incorporate lossless early stopping criteria. The resulting gains in computational efficiency allow the use of more complex measurement models to increase the map's accuracy, such as a novel angular uncertainty-aware measurement model. Validated with synthetic RGB-D and real-world 3D LiDAR datasets, the method has been shown to be effective in achieving high-quality mapping results with reduced memory and computational demands. In particular, the method is shown to achieve state-of-the-art recall on challenging obstacles such as thin surfaces and vegetation. Moreover, we demonstrate the system's flexibility by integrating multiple sensors into a single map with per-sensor resolution. This feature enables the use of a single map for tasks that would have

required dedicated maps in the past, offering a more efficient solution for autonomous robotic systems.

In the second research chapter, we presented an efficient, low-latency reactive obstacle avoidance method to safely navigate unknown or changing 3D environments. The approach is based on the idea that objects that are far from the robot do not have to be processed at the same level of detail as nearby obstacles for effective and safe navigation. We therefore propose to represent obstacles surrounding the robot using multi-resolution cells and introduce a hierarchical algorithm to extract such an abstraction from hierarchical volumetric maps efficiently. Using the framework of RMPs, we then attach one policy to each multi-resolution obstacle and combine them in parallel to obtain a highly efficient reactive collision avoidance and navigation system. Our numerical analyses demonstrate that lowering the resolution for distant obstacles allows the system to significantly expand its perceptive radius without losing accuracy. A key benefit of using RMPs is their high degree of modularity, which makes it easy to combine our reactive collision avoidance policy with other RMPs to satisfy additional objectives, such as goal-seeking, to create a comprehensive navigation system. Thorough evaluations, conducted across a range of indoor and outdoor maps, reveal that our system matches the performance of optimization-based planners such as CHOMP while achieving a $50\times$ reduction in planning time, removing the need for pre- or post-processing steps, such as ESDF generation or trajectory tracking control. Deployed on a real MAV powered by an Nvidia Jetson AGX Orin, the system is shown to successfully negotiate an indoor obstacle course and operates at $200\mathrm{Hz}$ with an end-to-end latency of $36\mathrm{ms}$ while only using 2.4 threads for mapping and planning.

The final research chapter introduced an efficient, accurate, and resolution-complete global planner. The fundamental building block for our algorithm is a multi-resolution representation that can be used to store the planner's intermediate solutions. By generalizing the representation used by any-angle planners such as Theta* [6] to multi-resolution and encoding it into an octree data structure, it combines high accuracy and memory efficiency. A multi-resolution algorithm that efficiently explores the search space and computes the shortest path in a coarse-to-fine manner, dynamically increasing the resolution only where needed, further improves runtime. Finally, a special initialization procedure closes the accuracy gap between our multi-resolution planner and Theta* without compromising efficiency. Extensive evaluations are performed across a variety of maps of real indoor and outdoor environments. Ablations are presented to quantify the properties of each core component of our method. We also compare our method to a representative selection of search and sampling-based planners. The results show that our multi-resolution planner empirically maintains the completeness guarantees of search-based planners running at the highest resolution.

6.2 Conclusions

Reflecting on the research objectives of this thesis, our study on multi-resolution representations yielded substantial improvements across several key areas. In terms of the first research objective, mathematical rigor, using wavelet decomposition allowed us to define a hierarchical representation whose properties are clearly defined and satisfy the MRA conditions, which eliminate the need for heuristics and guarantee correctness when the map is updated or queried in a coarse-to-fine manner.

The second research objective was to increase efficiency, which we addressed by capitalizing on multi-resolution representations and hierarchical algorithms. This approach significantly reduced memory and computational demands by leveraging the fact that real environments predominantly consist of free space. We selected Haar wavelets for their optimal balance between compact storage, efficient updates, and compatibility with efficient data structures, notably improving the scalability of volumetric mapping.

Our final two research objectives were to improve the flexibility and usability of volumetric maps. As shown in all three research chapters, multi-resolution allows for a granular trade-off between efficiency and accuracy. In Chapter 3, we first showed how, using wavelet decomposition, a single map can seamlessly be updated by multiple sensors with per-sensor resolution and queried at any resolution at any time, eliminating the need to maintain dedicated volumetric maps for different tasks. In Chapter 4 and Chapter 5, we then demonstrated how our map representation can be combined with hierarchical algorithms to enable accurate, reliable, and highly efficient global path planning and reactive navigation. In conclusion, the work presented in this thesis paves the way for more autonomous, versatile robotic systems that can reliably accomplish complex tasks in large and diverse environments.

All the methods developed in this thesis, including the *wavemap* framework and reference implementations of *waverider* and *wavefinder*, are released as open-source projects. When combined, these tools constitute a full navigation system. However, we believe the true potential of *wavemap* lies in its versatility and flexibility. The framework, therefore, includes a growing toolbox of general-purpose algorithms and extensive documentation. We look forward to seeing how researchers and practitioners will leverage and customize these tools to advance their autonomous systems.

6.3 Future Work

The findings of this thesis open up many opportunities for future research, some of which we highlight in the following.

High-Level Scene Understanding and Semantics

In order to reason about and interact with their environment, autonomous agents often need an understanding of their surroundings that goes beyond pure geometry. A wide range of formulations are currently in use to represent environments while incorporating high-level information such as semantics. One family of approaches divides the scene into submaps on a per-object or panoptic basis. These methods typically model the geometry within each submap using a general geometric representation, such as a fixed-resolution occupancy grid or TSDF. By instead using *wavemap* to model each submap's geometry, they could significantly increase their overall efficiency, scalability, accuracy, and flexibility. Another common approach is to model the entire scene using a single volumetric map and estimate the semantic labels per voxel. Note that closed-vocabulary semantic classes could directly be included in traditional hierarchical volumetric maps such as Octomap [4], for example, using bit masks. This would already be interesting, as it makes it possible to use semantics-aware hierarchical algorithms in downstream tasks. However, if closed- or perhaps even open-vocabulary semantic labels could be represented in a way that their numerical differences become meaningful, they could even be encoded using wavelet decomposition. This would be particularly interesting, as all of the advantages of *wavemap* would then readily apply to storing, updating, and querying semantic-volumetric maps.

Hierarchical volumetric maps as an input to learning-based systems

Learning-based approaches are increasingly outperforming classical methods across a wide array of robotic tasks, ranging from object segmentation to traversability estimation and locomotion policies. However, many learning-based systems are still limited to 2D inputs due to the complexity of processing 3D data. Static, fixed-resolution 3D grids are simple to use but require excessive amounts of memory and processing power to cover a reasonable volume at a sufficiently high resolution for most applications. Alternative methods, such as PointNet, can directly operate on sparse, unstructured 3D data but are inherently approximate. In contrast, hierarchical volumetric maps can efficiently represent 3D geometry at variable levels of detail without sub-sampling. Furthermore, many parallels exist between the wavelet transform and convolutional

neural networks. It would, thus, be interesting to investigate whether an encoder network can be designed to efficiently extract feature embeddings directly from *wavemap* for learning-based spatial perception or navigation tasks.

Spatio-temporal mapping

Spatio-temporal volumetric maps are often represented by splitting the world into submaps, each covering distinct time intervals. Although this approach is reasonably efficient in terms of resource usage, it can only capture the world at a very low temporal resolution. At the start of this thesis, we were drawn to wavelets due to their successful applications in 2D and 3D (medical) image compression. In a similar vein, many parallels exist between compression of spatio-temporal volumetric maps and (wavelet-based) video compression. It would, therefore, be interesting to combine the outcomes of this thesis with those from the video compression research community. With the goal of developing a method to efficiently store, update, and query high-resolution spatio-temporal volumetric maps.

Efficient and Accurate Deformations

Various applications call for efficient and accurate methods to deform volumetric maps. For example, to correct errors introduced by drifting pose estimates after detecting a loop closure. Existing solutions often solve this problem by de-integrating and re-integrating measurements to rebuild the parts of the map that require the largest amount of change, which is computationally expensive and requires all measurements to be stored. Another solution is to represent the world using a collection of movable, interpolated submaps. However, this approach can only simulate smooth deformations when the submaps are small and feature a high degree of overlap. This is also inefficient, as the amount of redundant information to store and interpolate increases linearly with the number of overlapping submaps. One way to overcome these limitations would be to continuously deform the volumetric map. Deforming volumetric models has received significant attention in the computer graphics community and shares significant similarities with the more general problem of (mesh) parameterization, i.e. smoothly mapping mesh or grid vertices to corresponding coordinates on a 2D or 3D texture. We believe that similar methods could successfully be applied to hierarchical volumetric maps and would significantly increase their efficiency, accuracy, and applicability within or in combination with SLAM systems.

Anytime mapping

The current version of *wavemap* integrates measurements into the map up to a fixed, sensor-specific, maximum resolution. By virtue of the coarse-to-fine integrator's early stopping criteria, the computational cost, therefore, drops significantly when updating a region where the map's occupancy estimates have already converged. For many applications, this is convenient since it saves energy and frees up computational resources for other tasks. However, for certain tasks, it would be better for map updates to take a fixed amount of time while dynamically adjusting the update resolution. In the spirit of anytime planning, we refer to this feature as anytime mapping. Some inherent benefits of this approach are that it relieves the user from choosing the resolution *a priori*, guarantees a fixed latency even when the robot transitions between confined and wide-open spaces, and naturally adjusts the resolution to the robot's velocity. Given *wavemap*'s flexibility, it would even be possible to prioritize specific parts of the scene – in a similar fashion to how humans concentrate on objects they manipulate or narrow passages they want to traverse.

Advancing scalability

Although *wavemap* already exhibits better scaling than the state of the art, substantial improvements are still within reach. Our open-source implementation currently needs about $1.6\mathrm{GB}$ of memory to reconstruct $140 \times 240 \times 20\mathrm{m}$ environments at a resolution of $5\mathrm{cm}$. On a high-end laptop CPU, it can integrate LiDAR and depth camera data up to a resolution of $5\,\mathrm{cm}$ and $1\,\mathrm{cm}$, respectively, at sensor rate. Optimizing the implementation further might be worthwhile for applications requiring extremely large, high-resolution maps and industrial users targeting low-cost or low-powered devices. For very large-scale applications, one interesting aspect of wavelet-encoded volumetric maps is that they are particularly well-suited for out-of-core processing. Wavelet decompositions allow maps stored on disk to be loaded into memory gradually, starting with a coarse approximation that can progressively be refined in regions of interest by only loading missing detail coefficients. Given that the wavelet transform is linear, orthogonal, and localized, this approach is not limited to read-only operations, and changes can just as efficiently be written back to disk. Another interesting property of the wavelet transform is its compatibility with fixed-precision arithmetic, which might benefit certifiability by eliminating lossy floating point types and operations. Finally, significant gains could be achieved using hardware acceleration, advanced data structures, fixed-precision low-width numerical types, and more advanced encoding schemes – bridging the gap to lossless image encoders such as JPEG 2000.

Distributed systems

Although this thesis primarily focused on single-robot systems, wavelet-based representations are particularly well-suited to distributed applications. The previous subsection highlighted several ways to further decrease *wavemap*'s memory usage, which has even bigger implications when maps have to be transmitted over low-bandwidth or unreliable networks. The ability to initially exchange maps at a coarse resolution and progressively refine regions of interest without transmitting redundant data would significantly improve the efficiency and reliability of multi-robot or cloud-aided systems. Progressive transmissions can be taken one step further by sorting the transferred bits individually in order of their significance. Encoders that exhibit this property are referred to as embedded coders [5], and wavelet decompositions are naturally amenable to this type of encoding. One final advantage we have not yet mentioned is that the linearity of the wavelet transform means that map changes can be communicated differentially and incorporated by the receiver by simply summing up the changes. This enables particularly efficient collaborative mapping.

Multi-resolution distance fields

Many similarities exist between the algorithms used for any-angle planning and ESDF generation. The key difference is that planners are typically initialized using a single start vertex, while ESDF generators are initialized using a set of surface vertices. Furthermore, any-angle planners need to handle the fact that the shortest path can bend around obstacles, while the ESDF's magnitude and gradient always point to the closest obstacle. Therefore, changing *wavefinder*'s initialization procedure suffices to turn it into a coarse-to-fine, multi-resolution ESDFs generator. Just like *wavefinder*, multi-resolution ESDFs can exploit the inherent sparsity of real environments to be significantly more efficient than their fixed-resolution counterparts. They also offer significant advantages in terms of flexibility since their accuracy can dynamically be adjusted to the task. Furthermore, ESDFs are commonly used as a building block for other geometric operations, such as computing generalized Voronoi diagrams. We thus believe their development is a promising avenue for future research.

Exploration planning

An important capability for robot autonomy in unknown environments is autonomous exploration. However, determining the most informative exploration trajectory based on a robot's incomplete and changing model of the world fundamentally has a very high computational complexity. To make this problem tractable, existing exploration planners typically narrow down the options they consider by focusing on the map's frontiers or randomly sampling a small subset of candidate viewpoints or paths. Only considering frontiers is restrictive in terms of the gain formulations and tasks that can be achieved, while random sampling discards a lot of the information embedded in the map and neglects its underlying structure. It would be ideal if next-best viewpoints, or even paths, could efficiently and deterministically be extracted from the map. Hierarchical algorithms can generally be used to significantly speed up a broad range of expensive geometric operations, such as the computation of Generalized Voronoi Diagrams. Furthermore, in Chapter 5, we saw how hierarchical representations facilitate efficient, accurate, and reliable global planning. Based on these insights, we believe the use of hierarchical representations and algorithms could significantly improve the consistency, reliability, and efficiency of exploration planning – making it a promising direction for future research.

Bibliography

[1] L. Schmid, V. Reijgwart, L. Ott, J. Nieto, R. Siegwart, and C. Cadena, "A Unified Approach for Autonomous Volumetric Exploration of Large Scale Environments under Severe Odometry Drift," *IEEE Robotics and Automation Letters*, p. 8, 2020. 5, 12

[2] 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," in *IEEE International Conference on Robotics & Automation*, 2022, p. 3306–3313. 5, 12

[3] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 227–234, 2020. 5, 12

[4] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, 2013. 7, 23, 24, 30, 31, 42, 44, 46, 96

[5] S. G. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Elsevier/Academic Press, 2009. 7, 15, 16, 17, 18, 24, 99

[6] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-Angle Path Planning on Grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010. 8, 10, 62, 64, 65, 80, 94

[7] V. Reijgwart, C. Cadena, R. Siegwart, and L. Ott, "Efficient volumetric mapping of multi-scale environments using wavelet-based compression," in *Robotics: Science and Systems*, 2023. 9, 12, 42, 44, 46, 58

[8] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian

optimization for motion planning," *International Journal of Robotics Research*, 2013. 9, 44, 51

[9] V. Reijgwart, M. Pantic, R. Siegwart, and L. Ott, "Waverider: Leveraging hierarchical, multi-resolution maps for efficient and reactive obstacle avoidance," in *IEEE International Conference on Robotics & Automation*, 2024. 10, 13

[10] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. 12, 44

[11] L. Gasser, A. Millane, V. Reijgwart, R. Bähnemann, and R. Siegwart, "Voxplan: A 3d global planner using signed distance function submaps," in *IEEE International Conference on Robotics & Automation*, 2021, pp. 7901–7907. 12

[12] I. Daubechies, *Ten Lectures on Wavelets*, ser. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1992. 15, 24

[13] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, Apr. 1983. 15, 24

[14] C. Loop, Q. Cai, S. Orts-Escolano, and P. A. Chou, "A Closed-Form Bayesian Fusion Equation Using Occupancy Probabilities," in *International Conference on 3D Vision*, 2016. 22, 23, 24, 26, 27, 28

[15] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016. 22

[16] R. Mur-Artal, J. Montiel, and J. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, 2015. 22

[17] J. Schönberger and J. Frahm, "Structure-from-Motion Revisited," in *Conference on Computer Vision and Pattern Recognition*, 2016. 22

[18] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Europena Conference on Computer Vision*, 2014. 23

[19] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets," *Autonomous Robots*, 2013. 23

[20] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3D modeling and tracking," *Journal of Visual Communication and Image Representation*, 2014. 23

[21] J. Behley and C. Stachniss, "Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments." in *Robotics: Science and Systems*, 2018.

[22] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *International Symposium on Experimental Robotics*, 2014.

[23] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic lidar fusion: Dense map-centric continuous-time slam," in *IEEE International Conference on Robotics and Automation*, 2018.

[24] T. Whelan, S. Leutenegger, R. Salas Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM Without A Pose Graph," in *Robotics: Science and Systems*, 2015. 23

[25] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially Extended KinectFusion," in *AAAI Conference on Artificial Intelligence*, 2012. 23

[26] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, 1989. 23, 28

[27] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1996. 23, 24

[28] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera," in *ACM Symposium on User Interface Software and Technology*, 2011. 23

[29] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-Board MAV Planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. 23, 24, 31

[30] O. Kähler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray, "Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices," *IEEE Transactions on Visualization and Computer Graphics*, 2015. 23

[31] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *European Conference on Computer Vision*, 2020. 23

[32] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance Fields without Neural Networks," in *IEEE / CVF Computer Vision and Pattern Recognition Conferenc*, 2022. 23

[33] S. O'Callaghan and F. Ramos, "Continuous occupancy mapping with integral kernels," in *AAAI Conference on Artificial Intelligence*, 2011. 23

[34] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-Time 3D Reconstruction at Scale Using Voxel Hashing," *ACM Transactions on Graphics*, 2013. 23

[35] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data," *Sensors*, 2022. 23

[36] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. Kelly, and S. Leutenegger, "Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping," *IEEE Robotics and Automation Letters*, 2018. 23, 30, 42, 44, 46

[37] E. Vespa, N. Funk, P. H. J. Kelly, and S. Leutenegger, "Adaptive-Resolution Octree-Based Volumetric SLAM," in *International Conference on 3D Vision*, 2019. 24

[38] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P. Alcantarilla, and S. Leutenegger, "Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning," *IEEE Robotics and Automation Letters*, 2021. 23, 24, 31, 39, 44

[39] D. Duberg and P. Jensfelt, "UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown," *IEEE Robotics and Automation Letters*, 2020. 23, 24, 39, 42, 44, 46

[40] M. Yguel, O. Aycard, and C. Laugier, "Wavelet Occupancy Grids: A Method for Compact Map Building," in *Field and Service Robotics*, 2006. 23

[41] A. Schaefer, L. Luft, and W. Burgard, "DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform," *IEEE Robotics and Automation Letters*, 2018. 23

[42] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 23

[43] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *European Conference on Computer Vision*, 2020. 23

[44] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-Blox: A Scalable and Consistent TSDF-based Dense Mapping Approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018. 24

[45] S. Fuhrmann and M. Goesele, "Fusion of Depth Maps with Multiple Scales," *ACM Transactions on Graphics*, 2011. 24

[46] M. Yguel, O. Aycard, and C. Laugier, "Update Policy of Dense Maps: Efficient Algorithms and Sparse Representation," in *Field and Service Robotics*, 2008. 30

[47] K. Museth, "VDB: High-Resolution Sparse Volumes with Dynamic Topology," *ACM Transactions on Graphics*, vol. 32, no. 3, 2013. 30, 37, 82

[48] L. Schmid, J. Delmerico, J. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena, "Panoptic Multi-TSDFs: A Flexible Representation for Online Multi-resolution Volumetric Mapping and Long-term Dynamic Scene Consistency," in *International Conference on Robotics and Automation*, 2022. 33

[49] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera lidar inertial extension to the newer college dataset," *arXiv preprint arXiv:2112.08854*, 2021. 34, 51, 77

[50] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast Direct LiDAR-Inertial Odometry," *IEEE Transactions on Robotics*, 2022. 34, 77

[51] M. Tranzatto, M. Dharmadhikari, L. Bernreiter, M. Camurri, S. Khattak, F. Mascarich, P. Pfreundschuh, D. Wisth, S. Zimmermann, M. Kulkarni, V. Reijgwart, B. Casseau, T. Homberger, P. De Petris, L. Ott, W. Tubby, G. Waibel, H. Nguyen, C. Cadena, R. Buchanan, L. Wellhausen, N. Khedekar, O. Andersson, L. Zhang, T. Miki, T. Dang, M. Mattamala, M. Montenegro, K. Meyer, X. Wu, A. Briod, M. Mueller, M. Fallon, R. Siegwart, M. Hutter, and K. Alexis, "Team CERBERUS Wins the DARPA Subterranean Challenge: Technical Overview and Lessons Learned," 2022. 38

[52] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 50–56. 42

[53] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian Motion Policies," *arXiv:1801.02854 [cs]*, 2018. 42, 46, 48, 49

[54] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404. 42

[55] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019. 44

[56] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, 2018. 44

[57] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, 2011. 44

[58] H. Oleynikova, C. Lanegger, Z. Taylor, M. Pantic, A. Millane, R. Siegwart, and J. Nieto, "An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *Journal of Field Robotics*, 2020. 44

[59] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021. 44

[60] M. Pantic, I. Meijer, R. Bähnemann, N. Alatur, O. Andersson, C. Cadena, R. Siegwart, and L. Ott, "Obstacle avoidance using raycasting and riemannian motion policies at khz rates for mavs," in *IEEE International Conference on Robotics & Automation*, 2023. 44

[61] S. Kambhampati and L. Davis, "Multiresolution path planning for mobile robots," *IEEE Journal on Robotics and Automation*, vol. 2, no. 3, pp. 135–145, 1986. 44, 61, 63

[62] W. Du, F. Islam, and M. Likhachev, "Multi-resolution A*," *ArXiv*, vol. abs/2004.06684, 2020. 44, 64

[63] D. M. Saxena, T. Kusnur, and M. Likhachev, "AMRA*: Anytime multi-resolution multi-heuristic A*," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3371–3377. 44

[64] N. Funk, J. Tarrio, S. Papatheodorou, P. F. Alcantarilla, and S. Leutenegger, "Orientation-aware hierarchical, adaptive-resolution A* algorithm for UAV trajectory planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6723–6730, 2023. 44, 45, 64

[65] K. Goel, Y. G. Daoud, N. Michael, and W. Tabib, "Hierarchical collision avoidance for adaptive-speed multirotor teleoperation," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2022, pp. 20–27. 45

[66] M. Mattamala, N. Chebrolu, and M. Fallon, "An efficient locally reactive controller for safe navigation in visual teach and repeat missions," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2353–2360, 2022. 50

[67] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, 2022. 57

[68] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959. 60

[69] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. 60, 63, 65, 80

[70] D. Chen, R. Szczerba, and J. Uhran, "A framed-quadtree approach for determining euclidean shortest paths in a 2-d environment," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 668–681, 1997. 61, 64

[71] A. Nash, S. Koenig, and C. Tovey, "Lazy theta*: Any-angle path planning and path length analysis in 3d," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, pp. 147–154, 2010. 62, 64, 80, 82, 85

[72] S. M. LaValle, *Planning algorithms*. Cambridge ; New York: Cambridge University Press, 2006. 63

[73] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 995–1001 vol.2. 63, 80

[74] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. 63, 80

[75] J.-Y. Lee and W. Yu, "A coarse-to-fine approach for fast path finding for mobile robots," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 5414–5419. 64

[76] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical path-finding." *J. Game Dev.*, vol. 1, no. 1, pp. 1–30, 2004. 64

[77] F. Hauer, A. Kundu, J. M. Rehg, and P. Tsiotras, "Multi-scale perception and path planning on probabilistic obstacle maps," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4210–4215. 64

[78] D. T. Larsson, D. Maity, and P. Tsiotras, "Information-theoretic abstractions for planning in agents with computational constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7651–7658, 2021. 64

[79] T. Uras and S. Koenig, "An empirical comparison of any-angle path-planning algorithms," p. 206–210, Sep. 2021. 64, 85

[80] M. Faria, R. Marín, M. Popović, I. Maza, and A. Viguria, "Efficient lazy theta* path planning over a sparse grid to explore large 3d volumes with a multirotor uav," *Sensors*, vol. 19, no. 1, p. 174, 2019. 64

[81] S. J. Russell, P. Norvig, and E. Davis, *Artificial Intelligence: A Modern Approach*, 3rd ed., ser. Prentice Hall Series in Artificial Intelligence. Upper Saddle River: Prentice Hall, 2010. 65

[82] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org. 80