

Imperial College London
Department of Computing

Autonomous Exploration and Object Reconstruction with an MAV

Sotirios Papatheodorou

27th May 2025

Supervisor: Prof. Stefan Leutenegger
Second supervisor: Prof. Andrew Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Statement of Originality

I confirm that I authored the work titled “Autonomous Exploration and Object Reconstruction with an MAV” independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. Furthermore I have adhered to the rules set out in the citation guidelines, I have documented all methods, data and processes truthfully and fully, and I have mentioned all persons who were significant facilitators of the work.

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a [Creative Commons Attribution-NonCommercial 4.0 International Licence](#) (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Abstract

Deploying autonomous robots outside a controlled environment is a challenging task. Robots must be able to gather information about and model their environment, determine their next course of action, and safely navigate to their goal. As such, autonomous exploration is an integral component of many higher-level tasks tackled by autonomous robots. Micro Aerial Vehicles (MAVs) are a very popular robotic platform for autonomous exploration due to their capability to move freely in three dimensions, allowing them to navigate a variety of environments.

While there is prior work in autonomous MAV exploration, it is by no means a solved problem. Improvements can be made in terms of exploration speed, scalability to large environments, and robustness to odometry drift. Maybe most importantly, the integration of autonomous exploration with some higher-level robotic task has not been extensively researched.

In this thesis we propose autonomous exploration methods for MAVs that address these challenges. A fast and efficient exploration algorithm, a hybrid between sampling-based and frontier-based exploration is presented. Additionally, the traditional geometric formulation of autonomous exploration is extended to incorporate a downstream task, in this case the high-quality reconstruction of objects of interest. Finally, a submap-based large-scale exploration method suitable for both depth camera and LiDAR sensors is introduced. These contributions have the potential to enable the use of autonomous MAV exploration in a wider variety of environments and tasks.

Acknowledgements

I would like to thank all the people who supported and helped me during my PhD.

I am especially grateful to Prof. Stefan Leutenegger for supervising my PhD and for his continuous guidance and advice. Stefan was always helpful, honest, and understanding, motivated me to do the best work possible and helped me focus on interesting research problems.

I would like to thank all the members of the Smart Robotics Lab at Imperial College London for making my time there so great. Thank you Chris for being a good friend and never making a day at the office boring. Thank you Nils for your friendship, for the time we shared inside and outside the lab and for everything I've learned from you. Thank you Dorian for being such a great friend and flatmate, for all the fun we had and for being there for me when I needed it. Thank you Dimos for all the support, help and valuable advice. Thank you Anna, Masha, and Binbin for making everyday life at the lab enjoyable and for our excellent collaboration.

A big thank you also to all the members of the Smart Robotics Lab at the Technical University of Munich for making the transition to another country so smooth and my stay there so nice. Many thanks especially to Sebastián and Simon B. for being so welcoming from the beginning and for being such good friends and collaborators.

The Dyson Robotics Lab at Imperial College London has been a very welcoming and friendly space. I want to thank Andy for fostering such a nice environment and all the researchers I met there, and especially Edgar and Kentaro for the great times we had together.

I would also like to thank Iosifina, Amani and, Sabine for their support in all kinds of matters outside of research. Little would get done without them.

I also want to thank my family and all my friends for their support before and during the PhD. Thank you Mike for so many years of friendship and for being there even when away. Thank you Vicky for helping me through difficult times and for always welcoming me to your home. Thank you Stefanos for being so understanding and putting up with me taking up your couch. Thank you Irma and Filippas for making me feel at home in London and for all the amazing time we spent together. Thank you so much Ioanna for always being there for me, in good times and bad, for helping me keep going and for being so patient and understanding all these years. Finally, I would

like to thank my parents for always urging me to learn, improve and be creative, and for their constant support in all my endeavours.

I couldn't have done this without you.

Sotiris Papatheodorou

Contents

1	Introduction	15
1.1	Motivation	16
1.2	Contributions	17
1.3	Thesis Structure	21
1.4	Funding	22
2	Literature Review	23
2.1	3D Mapping and Reconstruction	24
2.2	MAV Motion Planning	34
2.3	MAV Control	36
2.4	State Estimation and SLAM	37
2.5	Autonomous Exploration	39
3	Preliminaries	51
3.1	Notation	52
3.2	Pinhole Camera Model	52
3.3	Occupancy Mapping in supereight	53
3.4	Adaptive-resolution TSDF Mapping in supereight	55
3.5	Multi-resolution Occupancy Mapping in supereight2	57
3.6	OKVIS2 VI-SLAM System	58
3.7	Submapping using supereight2 and OKVIS2	59
3.8	Path Planning Using Informed-RRT*	62
4	Information-Driven Autonomous Exploration	64
4.1	Introduction	65
4.2	Related Work	66
4.3	Problem Formulation	67
4.4	Methodology	69
4.5	Evaluation	75
4.6	Conclusion	81
5	Semantic Object-Centric Exploration	82
5.1	Introduction	83

Contents

5.2	Related Work	84
5.3	Problem Formulation	86
5.4	Methodology	87
5.5	Evaluation	94
5.6	Conclusion	99
6	Navigation and Exploration with SLAM in the Loop	100
6.1	Introduction	101
6.2	Related Work	102
6.3	Problem Statement	104
6.4	System Overview	104
6.5	Global Frontier Computation	107
6.6	Evaluation	109
6.7	Conclusion	116
7	Conclusion	117
7.1	Summary of Contributions	117
7.2	Limitations and Future Work	118
	Bibliography	121
A	Appendix	141
A.1	Happy Accidents	141

List of Figures

2.1	A point cloud visualisation of a LiDAR scan.	25
2.2	An <i>OctoMap</i> at different octree depths.	26
2.3	<i>KinectFusion</i> input and reconstruction example.	27
2.4	Example of an ESDF and a projective SDF.	28
2.5	Comparison of radiance field mapping methods.	31
2.6	<i>MID-Fusion</i> system overview.	32
2.7	An MAV trajectory through a cluttered environment.	35
2.8	Photograph of a hexarotor MAV.	37
2.9	Frontiers of a 2D occupancy grid map.	40
2.10	2D visualisation of receding-horizon, sampling-based exploration. . .	44
2.11	Example of semantic scene completion in <i>SC-Explorer</i>	49
3.1	The <i>supereight</i> octree structure.	54
3.2	The <i>supereight</i> adaptive resolution TSDF octree structure.	55
3.3	The <i>supereight</i> integration scale selection.	56
3.4	<i>supereight</i> 3D reconstructions at various integration scales.	56
3.5	A slice through a <i>supereight2</i> occupancy map.	58
3.6	Overview of the <i>OKVIS2</i> VI-SLAM system.	59
3.7	The submapping pipeline using <i>supereight2</i> and <i>OKVIS2</i>	60
3.8	Visualisation of reference trajectory anchoring.	61
3.9	Examples of trees spanned by RRT-based algorithms.	63
4.1	Live, on-board map and high-resolution offline reconstruction.	65
4.2	Overview of the information-driven autonomous exploration method. . .	70
4.3	Sparse 360° raycasted depth map and corresponding entropy map. . .	74
4.4	Explored volume over time in the <i>Apartment</i> environment.	77
4.5	Explored volume over time in the <i>Maze</i> environment.	77
4.6	Top-down view of a 3D map of the <i>Maze</i> environment.	78
4.7	Map of the <i>Powerplant</i> environment and explored volume over time. .	79
4.8	Real-world experiment 3D map and photograph.	81
5.1	Individual object and environment reconstructions.	85
5.2	Overview of the object-centric autonomous exploration method. . . .	88
5.3	Visualisation of the exploration method and the 360° gain images. . .	93

List of Figures

5.4	Object reconstructions and corresponding ground-truth meshes. . . .	96
5.5	Explored volume and percentage of objects found over time.	96
5.6	Background and object reconstruction accuracy over time.	97
5.7	Background and object completeness over time.	97
5.8	Background and object observation distance evaluation.	98
5.9	Real-world experimental setup and object reconstructions.	99
6.1	3D reconstructions from the real-world experiments.	102
6.2	Overview of the submap-based autonomous exploration method. . . .	105
6.3	Visualisation of local and global frontiers.	109
6.4	Reconstruction evaluation in the <i>Depot</i> environment.	111
6.5	Explored volume over time in the <i>Depot</i> environment.	112
6.6	Exploration planner safety evaluation.	112
6.7	Photographs of the MAVs used for the two real-world experiments. .	114
6.8	Explored volume over time for the two real-world experiments. . . .	115
A.1	2spooky4me (2021)	141
A.2	Lollipop (2021)	142
A.3	Modern Architecture (2022)	142
A.4	Uninitialised (2022)	143

List of Tables

4.1	Experiment parameters.	76
4.2	Per-iteration computation time.	80
5.1	Simulated experiment parameters.	95
6.1	Simulated experiment parameters.	110
6.2	Reconstruction accuracy and completeness.	113
6.3	Computation time and memory usage.	113
6.4	Real-world experiment parameters.	115
6.5	Real-world experiment results.	116

List of Acronyms

CNN Convolutional Neural Network	31
CPU Central Processing Unit	
DoF Degree of Freedom	32
EKF Extended Kalman Filter	
ESDF Euclidean Signed Distance Field	28
FoV Field of View	115
GNN Graph Neural Network	33
GP-GPU General-Purpose computing on GPUs	16
GPS Global Positioning System	
GPU Graphics Processing Unit	
ICP Iterative Closest Point	39
IMU Inertial Measurement Unit	37
LIO LiDAR-Inertial Odometry	39
LLM Large Language Model	33
LO LiDAR Odometry	39
LQR Linear Quadratic Regulator	36
LVI-SLAM LiDAR-Visual-Inertial SLAM	39
LiDAR Light Detection and Ranging	16

MAV Micro Aerial Vehicle	16
MLP Multi-Layer Perceptron	43
MPC Model Predictive Control(ler)	36
NBV Next-Best-View	43
NeRF Neural Radiance Field	29
OMPL Open Motion Planning Library	63
PID Proportional Integral Derivative	36
QP Quadratic Programming	36
RAM Random Access Memory	
RGB-D RGB-Depth	
RGB Red Green Blue	
RMSE Root Mean Squared Error	112
ROS Robot Operating System	
RRT Rapidly-exploring Random Tree	35
SDF Signed Distance Field	26
SFC Safe Flight Corridor	35
SLAM Simultaneous Localisation and Mapping	20
TSDF Truncated Signed Distance Field	26
VI-SLAM Visual-Inertial SLAM	20
VIO Visual-Inertial Odometry	20
VI Visual-Inertial	
VLM Vision-Language Model	119

Introduction

Contents of Chapter

1.1	Motivation	16
1.2	Contributions	17
1.2.1	Paper I: Information-Driven Autonomous Exploration	17
1.2.2	Paper II: Semantic Object-Centric Exploration	18
1.2.3	Paper III: Navigation and Exploration with SLAM in the Loop	20
1.3	Thesis Structure	21
1.4	Funding	22

1.1 Motivation

There has been spectacular improvement in the capabilities of autonomous robots over the last years. The availability of affordable depth cameras and Light Detection and Ranging (LiDAR) sensors, as well as small form-factor computers with the capability for General-Purpose computing on GPUs (GP-GPU) have been instrumental in this improvement. They have allowed the creation of robotic platforms with accurate on-board sensing and sufficient computational resources for running demanding algorithms and machine learning models. As such, there is increased interest in the deployment of autonomous robots in a wider array of environments and for a broader set of tasks. In tasks such as search and rescue, industrial inspection or environmental monitoring, it is often desired for autonomous robots to be deployed in previously unknown or unmapped environments. Therefore, some form of autonomous exploration is an integral part of downstream robotic tasks.

The objective of autonomous exploration is for a mobile robot to create an, as complete as possible, map of its environment using its on-board sensors. The current state of the map is used by the robot to determine which locations should be visited next in order to gather more information about its environment and expand the map. This results in a process where the robot navigates in the environment, while continuously updating its map, and using the gathered information to determine where to go next. The goal of the robot's deployment is sometimes the creation of this map of the environment, so that it can be later processed offline, or used as a reference map for the deployment of other robots later on. However, it is often the case that the creation of the map is merely a requirement for successful completion of the robot's task. For example, in a search and rescue scenario, the mission goal is finding the locations of all humans in a given region in a timely manner. Thus the robot should traverse the environment, prioritising locations where humans are more likely be, while ensuring that all potential locations are checked, and providing the rescue team with a map they can use to plan the rescue operation. Consequently, there is interest in coupling fast and efficient autonomous exploration with a higher-level task, in order to allow using autonomous robots for a greater number of purposes.

Micro Aerial Vehicles (MAVs) are a very popular and versatile robotic platform for autonomous exploration research. The fact that they can freely move in three dimensions allows them to navigate a variety of environments, regard-

less of the terrain type. Thus, MAVs can operate both outdoors and indoors, in multi-floor buildings, large warehouses or forests. They are also capable of high velocities, allowing them to explore large regions quickly. However, MAVs also pose unique challenges for autonomous exploration algorithms. Due to their limited flight time, typically between 10 to 20 minutes, exploring the environment as quickly as possible is critical. In addition, taking full advantage of their capability for free movement in three dimensions requires both mapping and exploration planning algorithms that operate efficiently in three dimensions.

The objective of this thesis is to develop autonomous exploration algorithms that are suitable for deployment on-board MAV platforms and that allow using exploration as part of a higher-level robotic task. As a result, the methods presented focus on exploration speed and computational efficiency. Autonomous exploration is also coupled with high-quality reconstruction of objects of interest, thus providing semantic and object-level context to the resulting map, information that can be necessary for downstream tasks. One of the proposed exploration algorithms is extended to work with a submap-based mapping approach and the scalability of the resulting system to large environments is demonstrated. Finally, all proposed methods are evaluated not only in simulation but also on real-world MAVs, showcasing their applicability to scenarios requiring online and on-board computation.

1.2 Contributions

1.2.1 Paper I: Information-Driven Autonomous Exploration

Anna Dai¹, Sotiris Papatheodorou¹, Nils Funk, Dimos Tzoumanikas and Stefan Leutenegger (2020). **Fast Frontier-Based Information-Driven Autonomous Exploration with an MAV**. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Dai et al., 2020]

Context

For autonomous exploration using MAVs to be practical, fast and efficient exploration algorithms are needed, as described in Section 1.1. The classical approach to

¹Equal contribution. Anna Dai implemented the exploration planning, while Sotiris Papatheodorou provided advice and helped correct issues with the exploration planner, and performed the simulated experiments. Both authors worked equally for the real world experiments.

autonomous exploration is to guide the robot towards frontiers, that is, the boundaries between known free and unknown space. Frontiers indicate the regions that will expand the robot’s map of the environment. However, frontiers are typically computed using computationally expensive clustering methods. Sampling-based autonomous exploration algorithms have also been used in recent years. Candidate next poses are randomly sampled and the robot moves to the one with the highest utility. Sampling-based methods often require fewer computational resources than frontier-based methods. Nevertheless, they typically result in slower exploration times since the samples are not necessarily close to frontiers.

Contribution

The first contribution of this thesis is a novel MAV exploration strategy that is both fast and computationally efficient. The goal of the exploration strategy is the reduction of map entropy regarding occupancy probabilities, which is reflected in a utility function to be maximised. It achieves fast and efficient exploration performance with tight integration between the octree-based occupancy mapping approach, frontier extraction, and motion planning—as a hybrid between frontier-based and sampling-based exploration methods. The computationally expensive frontier clustering employed in classic frontier-based exploration is avoided by exploiting the implicit grouping of frontier voxels in the underlying octree map representation. Candidate next-views are sampled from the map frontiers and are evaluated using a utility function combining map entropy and travel time, where the former is computed efficiently using sparse raycasting. These optimisations along with the targeted exploration of frontier-based methods result in a fast and computationally efficient exploration planner. This method is described in detail in Chapter 4.

1.2.2 Paper II: Semantic Object-Centric Exploration

Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas, Christopher Choi, Binbin Xu and Stefan Leutenegger (2023). **Finding Things in the Unknown: Semantic Object-Centric Exploration with an MAV**. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Papatheodorou et al., 2023]

Context

For many practical applications, generating a purely geometric map of the environment can be inadequate. Instead, a map contextualised with semantic and object-level understanding of the environment is desired. While there are existing methods to perform detection of objects of interest during exploration, their goal is typically just to detect object instances, not to create high quality object reconstructions. For certain tasks, such as autonomous exploration and mapping for inspection, obtaining quality reconstructions is necessary for success.

Contribution

The second contribution of this thesis is an autonomous exploration method suitable to the task of both finding specific objects in unknown space and reconstructing them to a target level of detail. Therefore, the environment reconstruction pipeline is extended to not only create a background map, but also object-level and semantically fused submaps. Importantly, the previous objective function of uncovering as much free space as possible in as little time as possible is adapted with two additional elements. First, a maximum observation distance of background surfaces is required to ensure target objects are not missed by image-based detectors because they are too small to be detected. Second, an even smaller maximum distance to the found objects is required in order to reconstruct them with the desired accuracy. This is based on the observation that measurement noise in depth cameras is proportional to the distance from the measured surface. The proposed approach retains good exploration performance while consistently detecting more objects of interest. This method is described in detail in Chapter 5.

1.2.3 Paper III: Navigation and Exploration with SLAM in the Loop

Sotiris Papatheodorou², Simon Boche², Sebastián Barbas Laina and Stefan Leutenegger (2025). **Efficient Submap-based Autonomous MAV Exploration using Visual-Inertial SLAM Configurable for LiDARs or Depth Cameras**. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Papatheodorou et al., 2025]

Context

Performing autonomous MAV exploration outside of a controlled environment presents additional difficulties. Without the availability of a motion capture system or similar infrastructure for MAV pose tracking, on-board state estimation must be employed. Visual-Inertial SLAM (VI-SLAM) is generally preferable over plain Visual-Inertial Odometry (VIO) since it can account for the inevitable odometry drift by leveraging loop closures. Loop closures arise from revisiting previously observed regions of the environment and result in deformation of past estimated MAV states. Using a monolithic map in conjunction with Simultaneous Localisation and Mapping (SLAM) would require non-rigid map deformation for it to remain consistent with the estimated states upon loop closures. A simpler and more commonly used solution to this problem is a submapping approach, where local maps are anchored to MAV states and are rigidly transformed along with them on loop closures.

Contribution

The final contribution of this thesis is an autonomous exploration method suitable for safe deployment in a wide array of environments by leveraging SLAM and local submaps. The submapping approach allows retaining global consistency by applying loop closure corrections to the relative submap poses. In order to facilitate large-scale exploration, a scheme for efficient computation of global, environment-wide frontiers from local submap frontiers is proposed. A sampling-

²Equal contribution. Sotiris Papatheodorou implemented the frontier computation and exploration planning and performed the simulation experiments for the proposed method, while Simon Boche integrated them with the SLAM, and submapping systems and performed the simulation experiments for the baseline method. Both authors worked equally on the real-world experiments.

based next-best-view exploration planner leverages the global frontiers to compute exploration trajectories maximising an information gain over time utility. The proposed method also seamlessly supports using either a LiDAR sensor or a depth camera, making it suitable for different kinds of MAV platforms. This method is described in detail in Chapter 6.

1.3 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 provides an extended literature review of past and current research in autonomous exploration and related topics.

Chapter 3 introduces notation and definitions and provides brief overviews of existing systems and algorithms this thesis is based upon.

Chapter 4 describes a hybrid, frontier sampling autonomous exploration method driven by potential information gain. The method is extensively evaluated in simulation and deployed on a real-world MAV.

Chapter 5 presents an extended autonomous exploration pipeline whose goal is to not just uncover as much free space as possible but also detect objects of interest and reconstruct them in high quality. The proposed approach is evaluated in several simulation scenarios and on-board an MAV, showcasing its capability to function on a constrained platform with real-world data.

Chapter 6 introduces a submap-based autonomous exploration framework for large-scale environments that requires no prior infrastructure and can operate with both depth cameras and LiDAR sensors. It is compared against a state-of-the-art submap-based exploration method and deployed on two different real-world MAV platforms.

Chapter 7 concludes the thesis with a discussion of the presented contributions, their limitations and potential directions for future research.

1.4 Funding

The work presented in Chapters 4 and 5 was funded by the President’s PhD Scholarship from Imperial College London.

The work presented in Chapter 6 was funded by the Munich Institute of Robotics and Machine Intelligence (MIRMI) of the Technical University of Munich and the Munich Center for Machine Learning (MCML).

Literature Review

Contents of Chapter

2.1	3D Mapping and Reconstruction	24
2.1.1	Point Clouds	24
2.1.2	Occupancy Mapping	25
2.1.3	SDF Mapping	26
2.1.4	Radiance Fields	29
2.1.5	Semantic and Object-Level Mapping	31
2.2	MAV Motion Planning	34
2.3	MAV Control	36
2.4	State Estimation and SLAM	37
2.4.1	Visual-Inertial Odometry and SLAM	38
2.4.2	LiDAR-(Visual)-Inertial Odometry and SLAM	39
2.5	Autonomous Exploration	39
2.5.1	Frontier-based Exploration	40
2.5.2	Information Gain-based Exploration	42
2.5.3	Sampling-based Exploration	43
2.5.4	Hybrid Exploration	45
2.5.5	Semantically-guided Exploration	48

An autonomous exploration system consists of components from several areas of robotics. These components play a significant role in the capabilities and performance of the final exploration system. As a result in this chapter we present the most relevant work in 3D mapping and reconstruction, MAV motion planning and control, and SLAM, before presenting related work in autonomous exploration.

2.1 3D Mapping and Reconstruction

An integral part of an autonomous exploration system is a map of the environment. This map is necessary to determine which regions have been observed already and where the robot should go in order to explore more of the environment. Additionally, the map representation employed should allow planning collision-free robot paths in order to allow safe exploration. Finally, the map used for exploration planning may also be important for some downstream robotic task and so should represent all information necessary for this task. As such, the choice of the map representation used in an autonomous exploration system is one of the most important design decisions, since it determines where and how it can be used.

In this section we present an overview of relevant and recent work in 3D map representations and methods that are suitable for real-time operation on-board constrained platforms, such as mobile robots. We also discuss map representations that allow representing more than geometry, for example integrating semantic-level information.

2.1.1 Point Clouds

A point cloud is a collection of points in 3D space represented by their Cartesian coordinates $[x, y, z]^T \in \mathbb{R}^3$. They are the measurements produced by depth cameras and LiDARs and as such are commonly encountered in the context of 3D mapping and reconstruction.

A point cloud can serve as a useful 3D surface visualisation for humans, but it is often not that useful for autonomous robotic tasks. While a point cloud can represent the surfaces of obstacles, it does not allow distinguishing between *free* and *unknown* space, making it unsuitable for safe, collision-free path planning. Additionally, naively combining the point clouds from multiple consecutive measurements into a single, scene-wide point cloud map results in a map whose memory

requirements grow linearly with time, severely limiting its applicability to larger environments.

For these reasons, most autonomous robotic systems employ some kind of volumetric map representation that allows aggregating measured point clouds in an efficient manner, while also allowing for collision-free path planning.

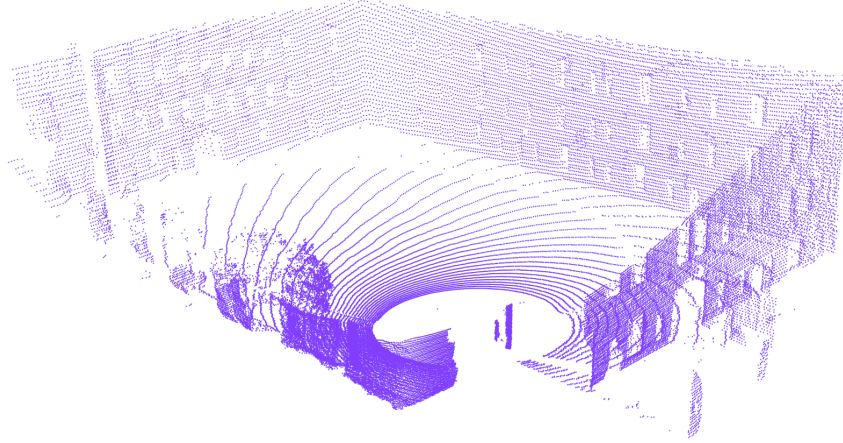


Figure 2.1: Point cloud visualisation of a LiDAR scan from the Newer College dataset [Ramezani et al., 2020].

2.1.2 Occupancy Mapping

Occupancy maps are a popular environment representation, especially for robotic applications. An occupancy map stores the occupancy probability of the environment, allowing for safe path planning by distinguishing between *free*, *occupied* and *unknown* (unmapped) space. Another advantage of the probabilistic nature of occupancy maps is that they can account for sensor measurement noise by using Bayesian updates.

A major milestone in 3D volumetric occupancy mapping was the release of *OctoMap* [Hornung et al., 2013], a framework using an octree to represent the environment in a memory-efficient manner. *OctoMap* performs pruning of leaf octree nodes with the same occupancy state, thus adapting the map resolution to the environment complexity. It also allows efficiently querying the map at multiple resolutions thanks to its underlying octree structure.

Several 3D occupancy mapping methods have been proposed since. *UFOMap* [Duberg and Jensfelt, 2020] builds upon *OctoMap* by explicitly mapping *unknown*

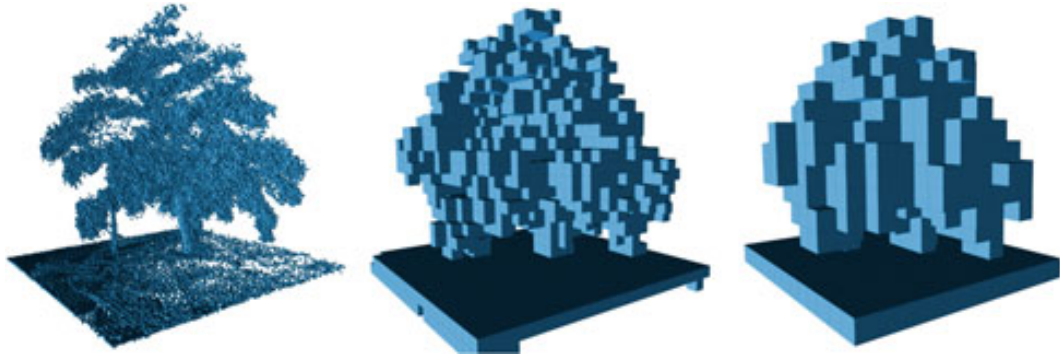


Figure 2.2: *OctoMap* of a tree queried at octree depths corresponding to voxel resolutions of 0.08 m [Left], 0.64 m [Middle] and 1.28 m [Right]. Images from [Hornung et al., 2013].

space and introducing more efficient update methods. *ColMap* [Fisher et al., 2021] reduces the memory requirements of occupancy mapping by aggregating similar voxels in columns, exploiting the fact that natural and human-made environments have a large amount of uniformity in the vertical axis. Another notable example is *supereight* [Vespa et al., 2018], which presents an octree-based framework that can be used for both Truncated Signed Distance Field (TSDF) and occupancy mapping. Its occupancy mapping capabilities were later improved in *supereight2* [Funk et al., 2021], achieving centimetre-level accuracy without sacrificing performance. We discuss *supereight* and *supereight2* in more detail in Section 3.3. Other works, such as *OHM* [Stepanas et al., 2022], improve the performance of occupancy mapping by using a GPU instead of a CPU, allowing efficient parallel processing of data.

2.1.3 SDF Mapping

3D volumetric mapping using Signed Distance Fields (SDFs) is a common choice for surface reconstruction. The seminal work of *KinectFusion* [Newcombe et al., 2011] uses an RGB-D sensor for real-time, accurate surface reconstruction by employing a TSDF map representation. However, real-time performance requires the use of a GPU, which is not available in all robotic platforms, or which might be needed for other tasks, such as running inference on deep learning models. More importantly, the map is represented as a fixed-size, densely allocated grid, limiting the method’s scalability to large scenes.

Several methods addressing these issues have been proposed since. In order to allow SDF mapping to scale to large environments, *Kintinuous* [Whelan et al.,

2012] integrates measurements into a sliding volume. Dense point clouds are extracted from regions of the SDF that exit the sliding volume and are incrementally integrated into a global mesh reconstruction. In [Zeng et al., 2013], an octree map representation is used to allow using a negligible amount of memory for empty parts of the environment, significantly reducing the memory required to map large scenes. Octrees are also successfully employed in methods that achieve high-performance SDF mapping on a CPU [Steinbrücker et al., 2014, Vespa et al., 2018], facilitating deployment in constrained robotic platforms. Another approach used for mapping larger environments using SDFs is spatial voxel hashing, as introduced by [Nießner et al., 2013] and later used by [Klingensmith et al., 2015] and *Voxblox* [Oleynikova et al., 2017]. A hash table is used to allocate and look up voxels or chunks of voxels only where the SDF should be updated, significantly reducing the memory requirements. In order to allow mapping of even larger scenes, while being more robust to imperfect pose estimates, *c-blox* [Millane et al., 2018] maintains a collection of *Voxblox* submaps. This is further expanded upon by *Voxgraph* [Reijgwart et al., 2019], which can also account for loop closures from a SLAM system by rigidly transforming the submaps.



Figure 2.3: Input point cloud [Left], surface reconstruction normals [Middle] and Phong-shaded surface reconstruction [Right] using *KinectFusion*. Images from [Newcombe et al., 2011].

ESDF Mapping

The previously mentioned SDF mapping methods use a projective SDF, meaning the distance used is along the measurement ray which can underestimate the true

distance to the closest surface, as would be encoded in a Euclidean Signed Distance Field (ESDF). This makes projective SDFs unsuitable for safe path planning, since they might encode erroneously large distances close to obstacles. While obtaining measurements from several different viewpoints will cause the SDF to approximate an ESDF, this often is not practical, especially on MAVs which have a limited flight time. A comparison between a projective SDF and an ESDF is shown in Figure 2.4.

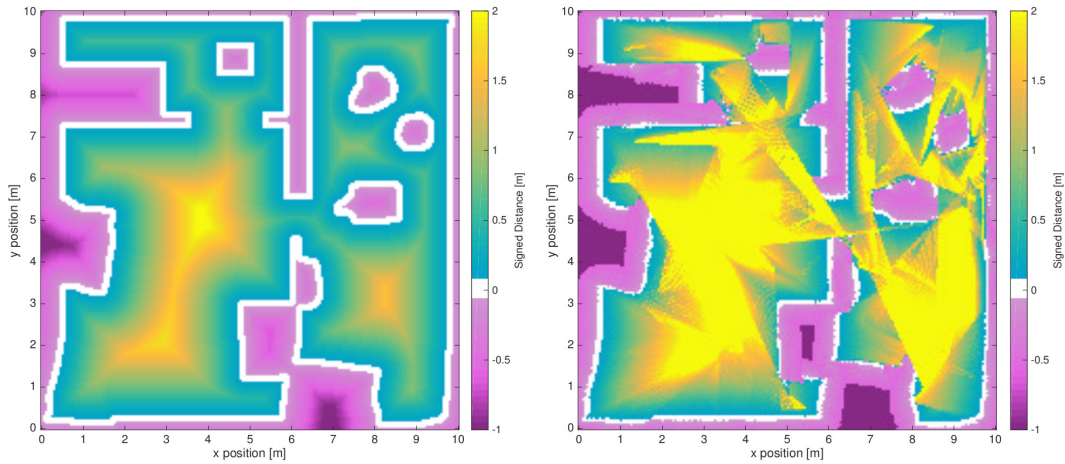


Figure 2.4: Visualisation of an ESDF [Left] and a projective SDF [Right] of a 2D environment. The reconstructed surface is shown in white, regions inside the surface are shown in pink and purple, while regions outside the surface are shown in green and yellow. Images from [Oleynikova et al., 2016b].

As a result, there has been interest in approximating an ESDF from a TSDF. *Voxblox* [Oleynikova et al., 2017] introduced a method for incrementally approximating an ESDF from a TSDF and used the result for MAV motion planning. Methods based on *Voxblox*, such as *c-blox* [Millane et al., 2018] and *Voxgraph* [Reijgwart et al., 2019] also inherit this capability. A faster and more accurate ESDF approximation algorithm is presented in *Voxfield* [Pan et al., 2022]. This is achieved by using more efficient bookkeeping and by employing a non-projective TSDF formulation which results in higher scene coverage. A fast and accurate incremental ESDF computation method is presented in *FIESTA* [Han et al., 2019]. It achieves high performance by employing specialised data structures that allow minimising the number of map queries during ESDF updates. It is agnostic to the underlying map representation, meaning it can be applied to both occupancy and TSDF maps. *VDB-EDT* [Zhu et al., 2021] is a memory efficient approach for generating an ESDF from an occupancy map, utilising a modified B+ tree for a hierarchical

representation of the ESDF.

2.1.4 Radiance Fields

In physics, radiance is the radiant flux emitted or reflected by a surface, per unit solid angle per unit area, while radiant flux is the energy of electromagnetic radiation per unit time. It is a directional quantity, since the radiant flux reflected by a surface depends on the direction from which it is observed. Radiance has been employed in the field of computer vision for photorealistic novel view synthesis in the form of radiance field. A radiance field encodes the direction-dependent radiance of some 3D space.

Neural Radiance Fields

The use of radiance fields for view synthesis was popularised by [Mildenhall et al., 2020], proposing the concept of Neural Radiance Fields (NeRFs). A NeRF represents the environment as a fully-connected, non-convolutional neural network. The network’s input is a 5D vector, consisting of a 3D position, and a view direction azimuth and elevation, and its output is an RGB colour and a volume density. Rendering the NeRF is performed by raycasting, querying the network at several points along each ray and accumulating the colour weighted by the volume density and transmittance. Since NeRF rendering is differentiable, the training (mapping) phase consists of updating the network weights using stochastic gradient descent, comparing ground-truth images with rendered view from the same poses. While the resulting NeRFs are memory efficient and allow rendering high-quality novel views, the training phase can take several hours and rendering is not real-time. Several approaches have been proposed since to address these challenges.

PlenOctrees [Yu et al., 2021] was the first method to achieve real-time rendering of NeRFs. First, a modified NeRF is trained that predicts spherical harmonic coefficients instead of a colour for each query point. The point colour can then be obtained by a weighted sum of the spherical harmonic bases given the view direction. This allows discretising the NeRF into a view-dependent octree, allowing the use of more efficient raycasting algorithms. While *PlenOctrees* achieves real-time rendering, NeRF training requires several hours.

This approach was taken one step further by *Plenoxels* [Fridovich-Keil et al., 2022], by completely removing neural networks from the pipeline. The scene is instead represented as a sparse 3D grid of spherical harmonics that can be directly optimised. This allows reducing the non-neural radiance field generation time to a few minutes, without loss of precision.

A breakthrough in NeRFs came with *InstantNGP* [Müller et al., 2022], presenting a method for NeRF training in a matter of seconds, while keeping rendering accurate and real-time. This was achieved by employing a multi-resolution hash encoding in combination with a small neural network, allowing parallelisation on modern GPUs.

The usage of NeRFs has not been limited to photorealistic rendering and novel view reconstruction. They have been used as part of visual localisation systems [Chen et al., 2024] or to provide an a-priori map in a VIO system [Katragadda et al., 2024]. Since NeRFs encode a volume density, they have been successfully used for path planning, either directly [Adamkiewicz et al., 2022], or by using them to approximate an ESDF [Pantic et al., 2022].

3D Gaussian Splatting

The splatting method for volumetric rendering was first proposed in [Westover, 1991]. Data samples are modelled as reflective, light-emitting, semi-transparent clouds and rendering is performed by accumulating the contributions of all clouds along each ray.

Splatting was first employed for radiance field representation and rendering in [Kerbl et al., 2023]. The representation consists of a set of 3D Gaussians, each defined by a position (mean), a covariance matrix, an opacity value and spherical harmonic coefficients. Due to the rendering being differentiable, stochastic gradient descent can be used to optimise the 3D Gaussians so that the error between the input and rendered images is minimised. The method achieves real-time, high-resolution novel view synthesis, while requiring less training time for higher quality renders than contemporary NeRF-based approaches.

Due to its high performance and rendering quality, 3D Gaussian splatting is starting to be used in several applications. A monocular SLAM system using 3D Gaussian splatting as the scene representations was presented in [Matsuki et al., 2024]. Camera tracking is formulated using direct optimisation against the 3D

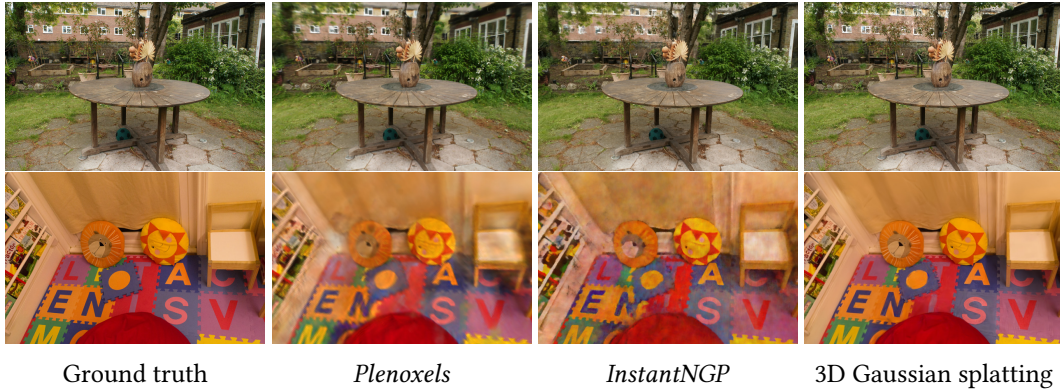


Figure 2.5: Comparison of radiance field mapping methods: *Plenoxels* [Fridovich-Keil et al., 2022] [Middle-left], *InstantNGP* [Müller et al., 2022] [Middle-right], and 3D Gaussian splatting [Kerbl et al., 2023] [Right]. Images from [Kerbl et al., 2023].

Gaussians and geometric regularisation is used to avoid artefacts due to incrementally constructing the map. It has been demonstrated that 3D Gaussian are also a suitable representation for path planning. In [Chen et al., 2025], level sets are extracted from the 3D Gaussians and used to generate safe polytope corridors, resulting in much lower planning times compared to NeRF-based navigation methods.

2.1.5 Semantic and Object-Level Mapping

Purely geometric environment maps are useful for safe navigation and certain robotic tasks such as inspection. However, they lack semantic information which is necessary for higher-level tasks.

With the advent of semantic segmentation Convolutional Neural Networks (CNNs) [Long et al., 2015, Noh et al., 2015], efficient, high-quality semantic mapping became a possibility. These CNNs receive an RGB image as input and produce an output image containing per-pixel semantic class predictions from a set of classes fixed at training time. In *SemanticFusion* [McCormac et al., 2017], the *ElasticFusion* [Whelan et al., 2015] SLAM system is augmented with a per-surfel probability distribution over the set of semantic classes, which is then updated in a Bayesian way. *Kimera* [Rosinol et al., 2020a] is a complete system that integrates VI-SLAM, collision avoidance, and high-quality, semantically annotated 3D meshing. While such approaches result in a map with semantic information, they are limited by the fact that there is no distinction between individual object instances,

at least not without some additional post-processing step.

Instance segmentation CNNs, such as *Mask R-CNN* [He et al., 2017], are able to predict semantic classes and also group pixels corresponding to the same object instance, simplifying the reconstruction of individual objects. *Fusion++* [McCor-mac et al., 2018] generates TSDF reconstructions of individual objects and stores them in an optimisable 6-Degree of Freedom (DoF) pose graph which is used for SLAM. *MID-Fusion* [Xu et al., 2019] presents dense a SLAM system capable of accurate reconstruction and tracking of multiple dynamic objects, thus increasing the state estimation accuracy. An incremental, instance-aware semantic mapping framework based on *Voxblox* [Oleynikova et al., 2017] is presented in [Grinvald et al., 2019]. A joint semantic-geometric segmentation method is employed to allow accurate segmentation of known classes, as well as the discovery of objects not part of the fixed set of classes the segmentation CNN was trained to detect. [Xu et al., 2022] proposes a learning-based method for 3D object completion which is applied to dynamic object tracking that is robust to occlusions and novel object views.

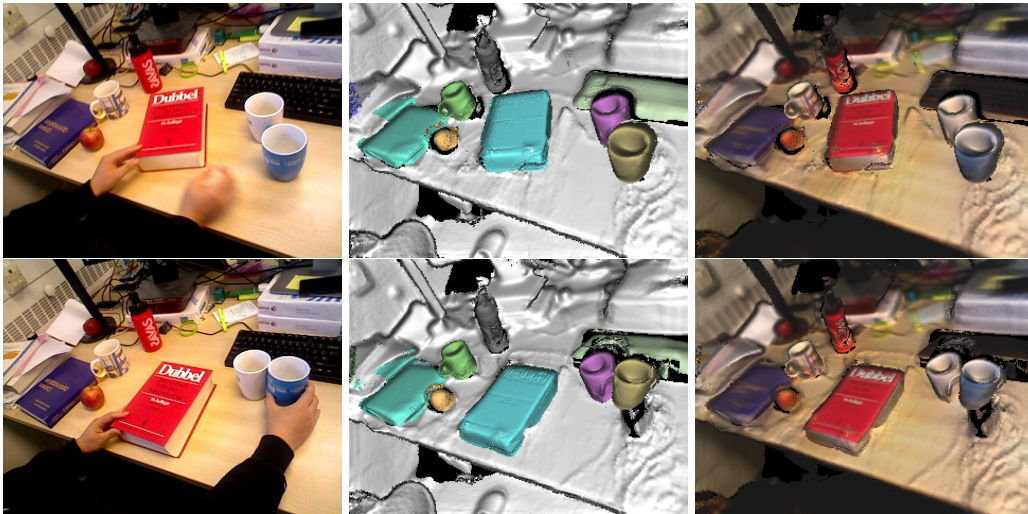


Figure 2.6: *MID-Fusion* system overview: input RGB images [Left], reconstructed objects and background [Centre], colour render of objects and background [Right]. Images from [Xu et al., 2019].

Object-level semantic information is not always sufficient for large-scale robotic tasks, which may require reasoning at the level of rooms or whole buildings. 3D scene graphs, proposed in [Armeni et al., 2019], aim to create a unified structure that models semantics at various levels as well as the relationships between them.

An automated method for generating 3D scene graphs is proposed, organising the scene in rooms containing objects by post-processing a 3D mesh of the environment and corresponding RGB panoramic images. The concept is extended in [Rosinol et al., 2020b] by allowing the inclusion of dynamic entities, such as humans and robots, while also proposing a method for constructing the scene graph directly from visual-inertial data. *SceneGraphFusion* [Wu et al., 2021] was the first real-time, incremental scene graph construction method. It utilises a Graph Neural Network (GNN) to predict a relationship graph from an incremental geometric segmentation and then fuse the result into the global scene graph.

The recent rapid advances in Large Language Models (LLMs) and foundation models have made it possible to overcome one of the main limitations of previous semantic mapping systems, the fact that they can only recognise a closed set of classes selected at training time. Two of the most influential and used foundation models in the field of semantic and object-level reconstruction are *CLIP* and *SAM*. *CLIP* (Contrastive Language-Image Pretraining) [Radford et al., 2021] consists of separate image and text encoders, trained so that they produce matching feature embeddings for each image-text pair in the training dataset. Thus, instead of being limited to a closed set of semantic classes like previous approaches, the semantic classes representable are only limited by the contents of the training dataset and the dimensions of the feature embedding. This results in a system that can reliably detect a much wider variety of semantic classes than previous works. *SAM* (Segment Anything Model) [Kirillov et al., 2023] a class-agnostic model for image segmentation. It can produce high quality masks even in images types outside its training domain and is not restricted to some fixed set of semantic classes.

Due to both *CLIP*- and *SAM*-based models not being limited to a fixed set of classes, they have been used extensively for semantic and object-level mapping. In *ConceptFusion* [Jatavallabhula et al., 2023], the environment is represented as a set of points, each of which has an associated feature vector. *SAM* is used to obtain a set of class-agnostic mask proposals, the input image is cropped around them and each individual crop is passed through the *CLIP* model, producing a feature vector. The per-pixel feature vectors are back-projected and fused into the respective map points. The *DINO* [Caron et al., 2021] and *AudioCLIP* [Guzhov et al., 2022] foundation models are used alongside *CLIP* to construct maps with multi-modal semantic information. *ConceptGraphs* [Gu et al., 2024] builds upon

ConceptFusion by employing an LLM, like *GPT-4* [Achiam et al., 2023], to generate a graph of object associations, providing a more structured scene representation that is useful for LLM-based task planning. *LEXIS* [Kassab et al., 2024] is a SLAM system that leverages *CLIP* to perform semantically-guided loop closures. The SLAM pose graph is augmented with *CLIP* features, which are used for incremental room detection, further refined through clustering. Semantically-guided loop closures are performed by comparing the *CLIP* features of the query image first with the per-room features and then with the individual keyframe features for the matched room. One great benefit of this approach is that it allows matching images with significant viewpoint variations, since the *CLIP* features are based on semantic and not visual information. Another notable system is *Clio* [Maggio et al., 2024], which receives a task specification in natural language and incrementally creates a 3D scene graph containing only information relevant to the task. It uses *FastSAM* [Zhao et al., 2023] and *CLIP* to generate a graph of 3D segments with associated semantic features, nodes of which are merged into objects based on a task-dependent metric. Further clustering is performed to group objects into places, thus creating a 3D scene graph. *Clio* takes advantage of the fact that *CLIP* encodes both text and images into the same feature space, allowing the assessment of the similarity between the task specification and the detected objects and places.

2.2 MAV Motion Planning

MAV motion planning is the process of computing a dynamically feasible and collision-free trajectory for the MAV to follow. Motion planning can be challenging due to the constraints imposed by the MAV dynamics, especially in the presence of obstacles, as is the case in most autonomous exploration scenarios. Smoother trajectories are more desirable since they require a smaller control effort, and thus energy consumption, resulting in longer flight times. In order to achieve smooth trajectories, a large number of methods represents trajectories as a series of polynomial motion primitives, computed for example through Bernstein [Gao et al., 2018a], B-spline [Usenko et al., 2017], or *MINVO* [Tordesillas and How, 2022] polynomial bases. A large portion of methods falls into one of two categories, gradient-based trajectory optimisation or trajectory optimisation in a

Safe Flight Corridor (SFC). A two-stage process is typically used in both cases, a path or trajectory initialisation, followed by an iterative spatiotemporal trajectory optimisation.

The first stage, path initialisation, consists of determining an initial collision-free path, typically using a spherical MAV approximation and with no regards to its kinodynamic feasibility. Nevertheless, the quality and feasibility of the initial path can have a great effect in the performance of both optimisation-based and SFC-based approaches. Sampling-based path initialisation typically employs some form of Rapidly-exploring Random Trees (RRTs) [LaValle, 1998], with improved convergence rate and sampling efficiency like Informed RRT* [Gammell et al., 2014], or taking kinodynamic constraints into account as in Kinodynamic RRT* [Webb and Van Den Berg, 2013]. Another option is to use a graph search method, often based on A* [Hart et al., 1968], which can be adapted to generate a kinodynamically feasible path [Dolgov et al., 2010], or take the MAV's orientation into account [Funk et al., 2023].

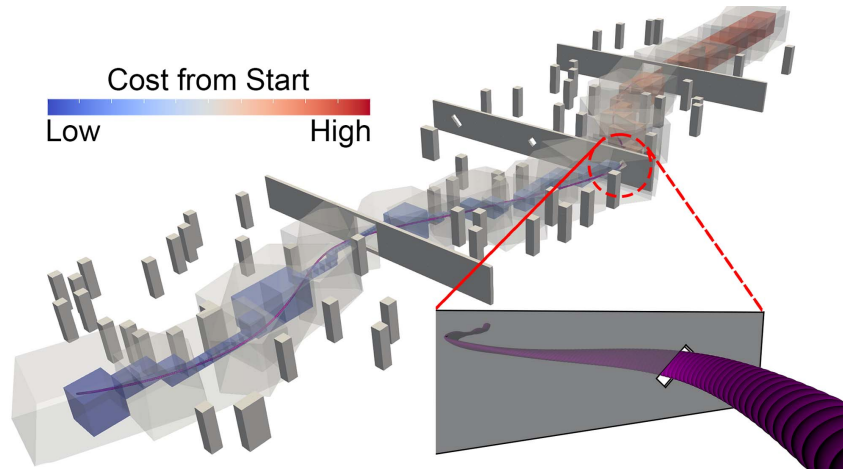


Figure 2.7: A polyhedral SFC and corresponding MAV trajectory through a cluttered environment. Image from [Funk et al., 2023].

Gradient-based trajectory optimisation approaches utilise an objective function that contains a collision cost and use the cost gradient along the trajectory to minimise the total cost. The produced trajectories can be discrete-time [Zucker et al., 2013] or continuous-time [Oleynikova et al., 2016a], while there are also approaches that don't require computing an ESDF to use for the cost [Zhou et al., 2020].

Trajectory generation methods based on SFCs use a more conservative free space approximation, modelling it as a set of convex solids such as cuboids [Chen et al., 2016], polyhedra [Gao et al., 2020], or spheres [Ren et al., 2022]. The convex solids act as inequality constraints in an optimisation problem and ensure the resulting trajectory remains safe.

2.3 MAV Control

Many MAV designs have been proposed over the years, ranging from quadcopters [Castillo et al., 2003], multirotor-MAVs with non-collinear motors [Brescianini and D’Andrea, 2016], designs allowing control over the thrust direction of individual motors [Kamel et al., 2018], or MAVs with adaptable morphology [Falanga et al., 2018]. In order to make the proposed method as universally applicable as possible, in this thesis we consider the simplest kind of MAVs, multirotors with collinear motors and fixed-pitch propellers.

Multirotors with collinear motors are underactuated systems, that is, they cannot be made to follow arbitrary trajectories in their 6-DoF configuration space. For example, only their position and yaw angle can be freely chosen while hovering, while their roll and pitch angles have to remain close to zero. These systems are controlled by varying the motor speeds, resulting in a change in thrust and torque. Early MAV works relied on linearised system models and used Proportional Integral Derivative (PID) or Linear Quadratic Regulator (LQR) controllers [Castillo et al., 2003, Bouabdallah, 2007]. One of the most influential MAV controllers was proposed in [Lee et al., 2010]. It performs tracking on SE(3) to avoid the singularity-related problems that earlier controllers had due to their use of Euler angles.

More recently, Model Predictive Control (MPC) has been employed for MAV control. MPC describes a class of discrete-time controllers that optimise the control inputs along a finite horizon to minimise state tracking error while satisfying input and state constraints. In the simplest case, a linear system model is used [Baca et al., 2016], typically resulting in a Quadratic Programming (QP) optimisation problem. A non-linear system model can also be used [Tzoumanikas et al., 2020a], allowing more accurate predictions and thus more aggressive flight. Another option is to switch between different MPCs using models linearised around different operating points [Alexis et al., 2012].



Figure 2.8: Photograph of the hexarotor MAV used in [Papatheodorou et al., 2023].

In this thesis we utilise the MPC presented in [Tzoumanikas et al., 2019]. It uses an MAV model linearised around the hover state and performs attitude control, meaning it requires a lower-level attitude controller to be present. It is relatively easy to obtain the MAV model, and tune the controller and it has proven to be robust in the real world.

2.4 State Estimation and SLAM

Performing autonomous exploration in environments without prior infrastructure, such as motion-tracking systems, requires estimating the MAV's state. For meaningful exploration planning in 3D, this state must include at least the MAV's position, and, if an omnidirectional sensor like a 360° LiDAR is not used, also its yaw angle. The MAV's controller requires estimating a larger state, consisting of the timestamped 6-DoF MAV pose and typically the linear and angular MAV velocities. Due to the capability of MAV platforms for highly dynamic manoeuvres, purely visual state estimation methods can lose track and fail. As a result we only present works that are designed to use a Inertial Measurement Unit (IMU), which directly measures linear acceleration and angular velocity, helping keep track even through aggressive flight.

The most common issue with state estimation systems is the accumulation of drift, that is, compounding estimation error due to measurement noise. This is most commonly tackled by employing Simultaneous Localisation and Mapping (SLAM). In SLAM, a map of the previously observed regions of the environment is retained and new observations are tested against this map to detect loop closures,

that is, revisits of previously observed regions. Loop-closures allow partially correcting drift by correlating new observations with old ones, where drift was lower. Although in principle a good exploration algorithm should not revisit regions, this is often impossible in practice due to the geometry of some environments. Exploring an environment with more than two dead-ends, common for building interiors, requires revisiting regions, irrespective of the starting position. As such, using SLAM is important for autonomous exploration, especially for large-scale missions where drift is larger and so we will focus on presenting works that perform SLAM.

2.4.1 Visual-Inertial Odometry and SLAM

Visual-Inertial Odometry (VIO) and Visual-Inertial SLAM (VI-SLAM) systems perform state estimation by using IMU measurements and images from one or more cameras. They are often classified into filter-based and optimisation-based approaches.

The seminal work of [Mourikis and Roumeliotis, 2007] introduced the multi-state constraint Kalman filter, a filter-based VIO method. It uses an extended Kalman filter to keep track of a number of past states and efficiently update them using 3D landmark observations by using a measurement model that does not require the landmarks to be part of the state. This approach has been the basis for several other methods, for example *ROVIO* [Bloesch et al., 2017], which uses image patches and photometric error as 3D landmark descriptors. There are also excellent open-source VIO implementations, such as *OpenVINS* [Geneva et al., 2020], improving upon the accuracy and computational efficiency of the original method while allowing widespread use. VI-SLAM systems have been built on top of filter-based VIO methods, for example *Maplab* [Schneider et al., 2018] which adds loop-closure capability to *ROVIO*.

Optimisation-based approaches usually formulate pose estimation as a factor graph optimisation problem, minimising residuals of IMU integration errors and visual reprojection errors. They tend to produce more accurate estimates than filter-based approaches but at a cost of increased computational requirements. One such system is the *OKVIS* [Leutenegger et al., 2015] VIO system, which uses a windowing approach to keep the optimisation problem tractable. It is built upon by *OKVIS2* [Leutenegger, 2022], which adds a visual loop closure mechanism. Another line of VI-SLAM systems is represented by *ORB-SLAM3* [Campos et al., 2021],

building upon [Mur-Artal et al., 2015] and [Mur-Artal and Tardós, 2017], opting to fix old states and ignore their corresponding estimate uncertainties. *VINS-Fusion* [Qin et al., 2019] is a VI-SLAM framework for fusing local sensors, such as cameras and IMUs, with global ones, such as GPS, achieving locally accurate and globally drift-free localisation.

2.4.2 LiDAR-(Visual)-Inertial Odometry and SLAM

Compared to cameras, LiDAR sensors have a higher field of view and are not affected by lighting and weather changes. As a result fusing LiDAR-based residuals to reduce the drift of VIO and VI-SLAM systems has been extensively investigated. An early influential LiDAR Odometry (LO) system is *LOAM* [Zhang and Singh, 2014], which uses geometric LiDAR-based feature residuals (edge and plane). These geometric features have been adopted by later works, such as the *LVI-SAM* [Shan et al., 2021] LiDAR-Visual-Inertial SLAM (LVI-SLAM) system. However, direct [Zheng et al., 2022] or Iterative Closest Point (ICP)-based [Vizzo et al., 2023] LiDAR-Inertial Odometry (LIO) methods have also proven to be effective. There are also hybrid approaches, such as the LVI-SLAM system presented in [Boche et al., 2024], where LiDAR measurements are fused into *supereight2* [Funk et al., 2021] submaps and the *OKVIS2* [Leutenegger, 2022] VI-SLAM system pose graph is augmented with submap-to-submap factors.

In practice the incorporation of LiDAR measurements into SLAM systems has tended to produce more accurate state estimates compared to plain VIO. According to the *KITTI* benchmark leaderboard [Geiger et al., 2012a, Geiger et al., 2012b], 34 out of the 50 methods with the lowest translation error make use of LiDAR measurements. The translation error ranges from about 0.5% to 1% for both vision-based and LiDAR-based methods.

2.5 Autonomous Exploration

Autonomous exploration consists of the problem of incrementally mapping an initially unknown environment using a sensor-equipped mobile robot. The main considerations in exploration algorithm design are a high exploration speed and an accurate environment reconstruction, which in turn is dependent on accurate localisation. The many approaches proposed over the years can be broadly cate-

gorised into frontier-based, information-theoretic, and sampling-based methods. There are also hybrid methods which combine aspects of multiple approaches to achieve even better results. The need to couple exploration with other downstream tasks has also given rise to semantically-guided exploration approaches that make use of more than geometric information.

2.5.1 Frontier-based Exploration

The concept of frontiers, the boundaries between known *free* and *unknown* space was first introduced in [Yamauchi, 1997]. Frontiers indicate regions that will expand the map of known space when observed. In [Yamauchi, 1997], adjacent frontier cells of the 2D occupancy map are clustered together and the robot moves to the closest frontier cluster. The environment map is updated incrementally and the frontier detection and selection process is repeated until no frontiers remain. The frontier clustering, while fast for 2D maps, can significantly slow down frontier computation in 3D, especially in high-resolution maps with lots of frontiers. The greedy frontier selection, like all greedy algorithms, can result in highly sub-

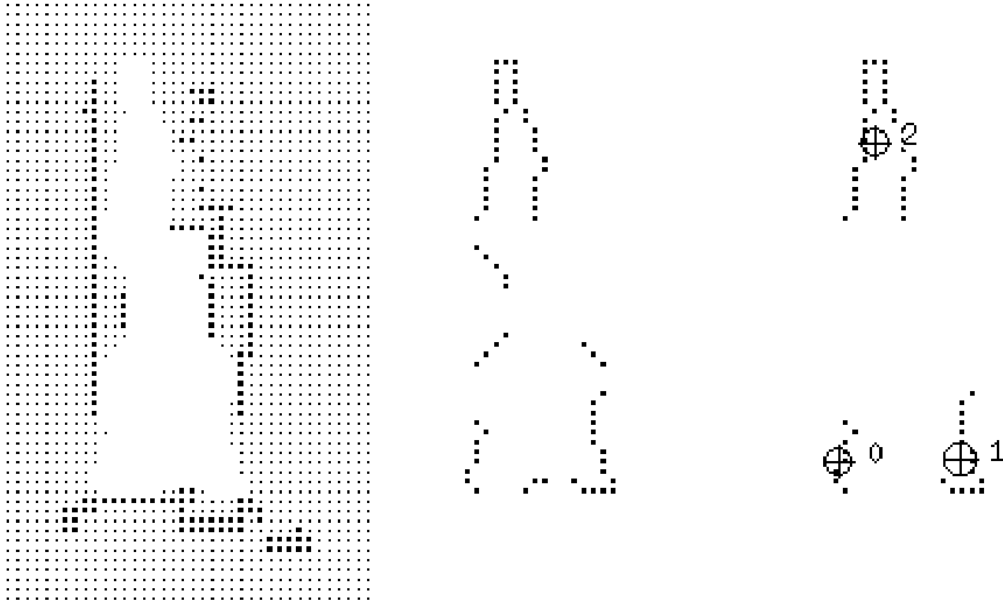


Figure 2.9: 2D occupancy grid map [Left] with large dots, small dots and white regions denoting *occupied*, *unknown*, and *free* space respectively. Extracted frontier cells [Centre]. Frontier cells clustered into groups and filtered [Right]. Images from [Yamauchi, 1997].

optimal exploration. More recent methods have attempted to tackle these shortcomings to achieve fast and computationally efficient exploration in 3D.

A less conventional frontier detection approach is presented in [Shen et al., 2012], based on the observation that there is a large number of erroneous frontiers in *free* space due to sensor occlusions and noise. Only the *occupied* parts of the environment are explicitly mapped, avoiding the memory-expensive dense *free* space mapping in 3D. *Free* space is instead modelled as a set of particles with dynamics governed by a stochastic differential equation. Collisions between the particles and *occupied* space are simulated and frontiers are detected based on the particle dispersion. After the frontiers are visited, the particles are re-sampled so that *free* space has a uniform sample density. While the method allows for computationally efficient exploration in 3D, the underlying assumption that uncluttered regions of the map correspond to unexplored space limits its utility in large-scale or outdoor environments.

Advances in 3D volumetric mapping have made it possible to efficiently map *free* space in 3D. The *OctoMap* framework, described in Section 2.1.2, is employed in [Zhu et al., 2015]. Frontiers are computed incrementally, based on the regions of the map that were updated, and clustered. The target frontier cluster is selected based on a simple estimate of the *unknown* volume around the cluster centre and its distance from the MAV.

In [Cieslewski et al., 2017] priority is given to maintaining a high MAV velocity during the exploration task by employing a two-tier planner. Frontiers within the MAV's current field of view are preferred since they require less control effort to reach and as a result allow retaining a high velocity. These frontiers are efficiently computed by post-processing the input depth images, avoiding expensive map lookups. If no frontiers exist in the field of view, the planner falls back to selecting the closest global frontier, computed by running the Dijkstra algorithm [Dijkstra, 1959] on the whole map.

The 2D exploration method presented in [Gao et al., 2018b] uses a topological graph for frontier tracking and selection. At each planning iteration the 10 lowest-cost frontiers are added to the graph and the selected one marked as visited. This scheme allows quickly searching the graph for the next frontier to visit when the robot needs to backtrack.

A thorough comparison of regular grid and octree-based occupancy maps for

autonomous exploration is presented in [Faria et al., 2019]. The comparison is performed using a global frontier computation and the Lazy Theta* [Nash et al., 2010] algorithm for path planning. It is shown the voxel aggregation performed by the hierarchical octree structure significantly reduces the time needed for both frontier computation and path planning.

Another notable frontier-based 3D exploration method is *FUEL* [Zhou et al., 2021]. Efficient frontier computation from a volumetric occupancy map is achieved through the use of an incrementally updated frontier information structure, which consists of frontier clusters and travel costs between them. A global coverage tour of the current frontier clusters is computed and the resulting path is further refined by considering viewpoints that observe a large volume of *unknown* space. *FALCON* [Zhang et al., 2024] builds upon *FUEL* by using a two-tier space decomposition that helps generate exploration paths that result in minimal revisits of already mapped regions.

2.5.2 Information Gain-based Exploration

Another group of autonomous exploration works use information-theoretic methods to determine which regions of the environment to explore next.

One of the early information-based exploration methods, [Bourgault et al., 2002], accounts for both localisation accuracy of the EKF-based SLAM system and the accuracy of the 2D occupancy grid. This is achieved by constructing an information-gain metric based on the uncertainty of the SLAM landmark positions, the vehicle pose and the occupancy map. The vehicle is guided towards local maxima of the information gain field, which is in turn updated as new measurements are available. A more recent method using a 2D occupancy grid is presented in [Bai et al., 2016], using a repeatedly trained Gaussian process to suggest informative next positions.

In [Kaufman et al., 2018], mapping is performed in 3D while exploration and motion planning are done in 2D to reduce the computational complexity. This choice is based on the observation that human-made structures have little variation in the vertical axis, but limits the method’s applicability to multi-storey buildings and natural environments. For the purpose of exploration planning, relevant features of the 3D map are projected to a 2D occupancy map and its Shannon entropy is used to determine informative next positions.

A real-time capable information-theoretic 3D exploration method is proposed in [Tabib et al., 2016]. It operates by constructing a future trajectory tree based on some motion primitives and computes Cauchy-Schwarz quadratic mutual information at each state. The tree is only extended to a finite horizon and similar states are pruned in order to keep the problem tractable.

A different approach to making information gain computation in 3D exploration tractable is described in [Palazzolo and Stachniss, 2018]. Instead of computing globally optimal next views, a greedy approach is followed. The computational requirements are further reduced by only considering next views on an adaptive hull that surrounds the region to be reconstructed. Emphasis is placed on generating smooth paths that are easier for a human operator to predict.

A more recent method for informative Next-Best-View (NBV) selection for reconstruction is based on a NeRF representation [Zeng et al., 2023]. The normally expensive information gain computation time is reduced to less than a millisecond by training a Multi-Layer Perceptron (MLP) to predict it, similarly to how a NeRF operates. This makes it possible to use a graph search algorithm like A* [Hart et al., 1968] to compute informative paths.

2.5.3 Sampling-based Exploration

Sampling-based exploration was pioneered by *RH-NBVP* [Bircher et al., 2016], showing that the NBV method [Connolly, 1985] can be applied to autonomous exploration in a receding-horizon manner. The core idea is to sample candidate next views in *free* space and evaluate each one based on some utility, selecting the candidate with the highest utility as the next goal. This strategy avoids the map-wide computation of frontiers but requires the evaluation of a utility function at each candidate. Thus the computational requirements of sampling-based approaches scale with the number of samples and computational complexity of the utility. In [Bircher et al., 2016] an RRT is sampled in the *free* subset of $\mathbb{R}^3 \times [0, 2\pi)$, the space of positions and yaw angles. Each node of the tree is a candidate next view whose utility is computed based as the number of visible *unknown* voxels discounted by the path length. The MAV flies to the first node along the path towards the node with the highest utility and then re-plans. Thus the MAV typically doesn't complete a full exploration path and instead re-plans often, causing it to get stuck in local minima. Another limitation of the approach is the fact that sam-

pling in a 4D space is inherently inefficient, requiring a large number of samples for efficient exploration.

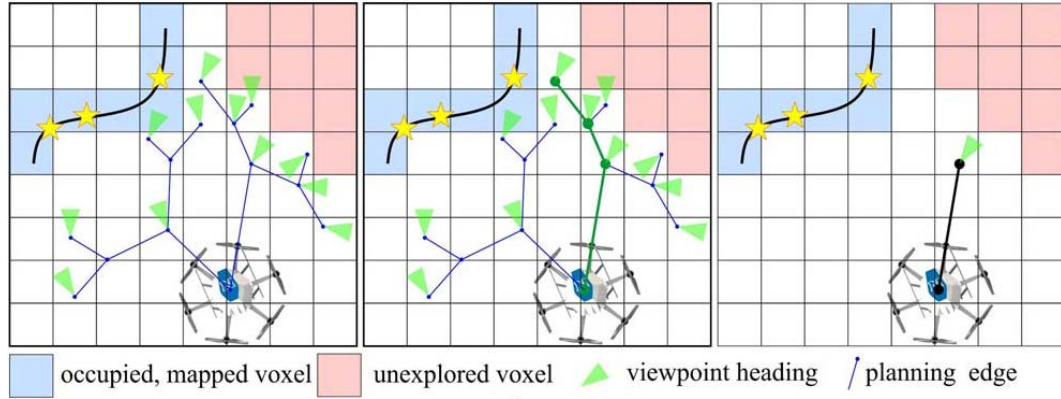


Figure 2.10: 2D visualisation of the *RH-NBVP* [Bircher et al., 2016] method. An RRT is sampled in *free* space [Left]. The leaf node with the highest utility is selected as the next goal [Centre]. Only the first segment of the goal path is tracked before re-planning [Right]. Images from [Papachristos et al., 2017].

Nevertheless, *RH-NBVP* [Bircher et al., 2016] has proven to be a very useful basis for future developments. It was extended to autonomous inspection tasks in [Bircher et al., 2018] by using a surface observation gain. A further improvement was proposed in [Witting et al., 2018], where a history of visited locations is maintained to avoid getting stuck at local minima. In [Papachristos et al., 2019] *RH-NBVP* is modified by increasing the view sampling bounds as the mapped volume grows, resulting in more efficient planning in underground mines consisting of long, branching tunnels.

An informative path planning algorithm inspired by RRT* is described in *Active3D* [Schmid et al., 2020]. A single tree of candidate next trajectories is continuously expanded and refined in *free* space, allowing the algorithm to achieve global coverage and compute path utility using global information. This is combined with a novel TSDF-based 3D reconstruction gain that takes measurement and localisation errors into account to achieve more accurate surface reconstruction.

UFOExplorer [Duberg and Jensfelt, 2022] is a sampling-based method designed to operate well large environments by avoiding map-wide sampling. *UFOExplorer* maintains a dense graph-based planning structure in *free* space that is incrementally updated when measurements are fused in the map. Each graph node is as-

sociated with an information gain and a path is planned to the closest node with a gain above a threshold using the multiple-goal A* algorithm [Hart et al., 1968]. While the goal selection is greedy, as in [Yamauchi, 1997], the fact that all structures used are incrementally updated and retained between planning iterations results in very fast exploration and path planning times.

A similar graph-based planning structure is used in [Akbari and Bernardini, 2021] to perform exploration of subterranean environments. The candidate utility function used takes the distance from obstacles into account, in addition to information gain and path length, in order to avoid collisions.

GS-Planner [Jin et al., 2024b] is a new sampling-based exploration method that maps the environment using 3D Gaussian splatting. Since Gaussian splatting only retains information for the occupied part of a scene, a voxel map is used to explicitly represent unknown space. The use of Gaussian splatting allows for efficient and more detailed rendering of gain images from novel views. Opacity-dependent safety bounds are computed for each Gaussian, making it possible to formulate collision-free MAV trajectory planning as an optimisation problem.

2.5.4 Hybrid Exploration

Some of the most performant and successful autonomous exploration methods combine elements of frontier-based, information-theoretic, and sampling-based methods to complement their strengths and address their shortcomings.

The work in [Stachniss et al., 2005] takes inspiration from both [Yamauchi, 1997] and [Bourgault et al., 2002], creating a system that uses frontiers to guide 2D exploration while also considering localisation uncertainty. A Rao-Blackwellized particle filter is used to model the uncertainty of the robot trajectory and the dependence of the map uncertainty on it. The expected information gain is computed for a finite set of actions, exploration actions that visit frontiers, and re-visiting actions that trigger loop closures and improve localisation accuracy.

The approaches of [Charrow et al., 2015a, Charrow et al., 2015b] combine several exploration methodologies to achieve fast exploration using both ground and aerial robots. Trajectories to candidate next poses sampled close to frontier clusters are first computed and then an objective function using Cauchy-Schwarz quadratic mutual information is used to select the most informative trajectory.

The method described in [Senarathne and Wang, 2016] is based on the observation that surface frontiers can often be as useful as 3D frontiers. Surface frontiers are boundary voxels of the *occupied* space surface that are next to *unknown* space. Searching for frontiers at the boundaries of a 2D surface in 3D space is more computationally efficient than searching in 3D space. Candidate next views are sampled close to surface frontiers and evaluated based on length of the visible frontiers and the estimated cost to reach them. While this method results in faster exploration planning, the focus on viewing surfaces can result in less *free* space being discovered, which can limit the space available for safe path planning.

AEP [Selin et al., 2019] is another influential method that builds upon *RH-NBVP* and proposes several improvements. Sparse instead of dense raycasting is used to estimate the *unknown* volume of candidate next views, allowing fast operation at much finer map resolutions. The sampling efficiency is greatly improved by only sampling 3D positions instead of positions and yaw angles. The yaw angle of each candidate is then computed by performing sparse 360° raycasting and selecting the yaw angle with the highest gain. To prevent *RH-NBVP* from getting stuck at local minima it is paired with a frontier-like global exploration planner. Given the observation that the potential candidate gain is a continuous field in $\mathbb{R}^3$, it is modelled as a Gaussian process constructed from the cached gains computed by raycasting. This allows computing the potential gain of some of the RRT nodes spanned by *RH-NBVP* by interpolating the Gaussian process, avoiding the more expensive raycasting. RRT nodes with a high gain are cached and act as frontiers, without the need for an explicit frontier computation. All these improvements result in faster and more computationally efficient exploration planning compared to *RH-NBVP*.

An exploration planner for large-scale surface reconstruction is presented in [Kompis et al., 2021]. It uses an NBV planner that samples candidate next views close to surface frontiers since the goal is to reconstruct surfaces and not volumes.

A group of methods improve the candidate next view sampling efficiency by sampling exclusively close to frontiers. In [Zhong et al., 2022] frontiers of the 3D occupancy map are incrementally detected and clustered to avoid operations on the whole map. Path planning to the candidates is made more efficient by using a multi-goal planner and a smooth goal path is generated by performing B-spline trajectory optimisation. *ECHO* [Yu et al., 2023] uses a heuristic designed to avoid

revisiting the same areas to select the best candidate view, which is subsequently refined to increase safety and reduce energy consumption. In [Zeng et al., 2024] a NeRF-based and an occupancy map are maintained in parallel, the latter used to detect frontiers. Candidate viewpoints are sampled closed to the frontiers and are evaluated based on the expected reduction in the NeRF uncertainty.

Osprey [Border et al., 2024] is a complete LiDAR-based aerial system for multi-session autonomous industrial inspection. It uses a LiDAR-Inertial SLAM system with geometric loop closures to obtain accurate poses for the collected data. The next MAV pose is determined by an NBV approach that considers both surface frontiers, expanding the mapped surface, and achieving a minimum measurement density on surfaces, important for high-quality reconstructions. Both the SLAM and exploration systems can store and reload their state to allow mapping of large structures which require multiple flights.

GBPlanner [Dang et al., 2019] is a hierarchical exploration planner tailor-made for large-scale subterranean environments, consisting mainly of long, branching, narrow passages. The local planner spans an RRT and optimises for paths that explore the local region while maintaining as large a clearance as possible from obstacles. The global planner incrementally builds a graph over the entire explored space and keeps track of vertices located near global frontiers. This allows the MAV to quickly move to another frontier upon reaching a dead-end and to return to the home position once the mission is finished.

In order to allow exploration of large environments in the presence of odometry drift *GLocal* [Schmid et al., 2021] utilises a submapping strategy and a hierarchical exploration planner. The global map consists of a series of periodically generated submaps based on *Voxgraph* [Reijgwart et al., 2019], the poses of which are incorporated in a pose-graph optimisation to allow handling loop closures. Once a submap is completed it is skeletonised using the method presented in [Oleynikova et al., 2018b], creating a traversability graph. The skeletons of overlapping submaps are connected at query time, allowing global paths to be found using a graph search algorithm like A* [Hart et al., 1968]. Local exploration planning is performed on a sliding-window map using an RRT*-based planner, similarly to *Active3D* [Schmid et al., 2020]. Once the utility of the immediate region becomes low, exploration switches to the global planner which guides the MAV towards global frontiers.

2.5.5 Semantically-guided Exploration

The autonomous exploration methods presented so far follow a purely geometric approach. This is the case for both their goals, reconstruction of a structure for inspection for example, and in how they select which regions should be observed next, typically using some direct or indirect metric of map completeness or accuracy. Incorporating semantic information in exploration planning allows performing more high-level tasks, for example finding a fridge, and can also be used to better guide the robot towards the goal, since a fridge is much more likely to be in a kitchen.

Autonomous exploration is combined with object search in [Dang et al., 2018a]. This is achieved by modifying the candidate view gain of the *RH-NBVP* pipeline to also reward high-resolution object re-observations. Semantic linking maps are introduced in [Zeng et al., 2020] as a probabilistic method of modelling and updating inter-object spatial relationships. The inter-object relationships are then used by the NBV planner to favour views that are likely to contain the object of interest. Reinforcement learning has also been used for object search, as in [Chaplot et al., 2020] where a navigation policy network is trained. It incrementally constructs top-down maps by projecting RGB-D and semantic images and then uses these maps in the policy network to select the next goal.

In [Ashour et al., 2020] the goal is to autonomously create a semantically-annotated occupancy map of the environment. The utility of the sampling-based exploration strategy balances between exploring unmapped regions and producing high-confidence semantic annotations in the existing map. A Bayesian multi-class semantic mapping system is introduced in [Asgharivaskasi and Atanasov, 2021]. It is capable of evaluating many potential robot trajectories online thanks to an efficient computation of a lower bound for the Shannon mutual information between the map and the measurements.

In [Lu et al., 2024] the goal is to generate an occupancy map of the environment that contains semantic annotations distinguishing between the background and objects of interest. Exploration is performed by a receding-horizon planner that considers geometric uncertainty, semantic class uncertainty and the ground robot’s kinematic constraints.

Star-Searcher [Luo et al., 2024] is a system for autonomous exploration and object search. The occupancy map is augmented to also track the minimum distance

voxels have been observed from, similarly to [Papatheodorou et al., 2023], keeping track of regions that have not yet been inspected from the desired distance. Candidate next views are sampled close to frontier clusters and uninspected regions, and ranked by a utility function that takes into account the number of unknown and uninspected voxels as well as the relative angle surfaces are observed from. A history-aware global planner computes a tour of the viewpoint clusters that avoids revisiting the same regions and a local planner generates a viewpoint tour for each cluster while taking the MAV kinodynamic constraints into account.

Semantic information can also help speed up geometric exploration as demonstrated by *SC-Explorer* [Schmid et al., 2022]. It uses a semantic scene completion network to predict 3D scenes from depth images. The 3D scene predictions are then incrementally fused into a separate map layer that is aligned with the TSDF map. An exploration planner based on *Active3D* [Schmid et al., 2020] is used to guide the MAV towards region where the scene predictions are more uncertain. This results in faster exploration since more informative paths are selected and small, easy to complete frontiers are ignored.



Figure 2.11: Example of semantic scene completion in *SC-Explorer*. Top-down view of the TSDF reconstruction with predicted voxels in orange, cyan, and green. The goal path shown in green is towards a region with high completion uncertainty. Image from [Schmid et al., 2022].

STAIR [Jin et al., 2024a] is an active implicit reconstruction pipeline that uses semantic information to reconstruct object classes of interest in greater detail. The NeRF used for reconstruction is augmented to also encode semantic class informa-

tion from input segmentation images. This allows considering the reconstruction uncertainty only in the regions where objects of interest are located, selecting next views that will result in reconstructing them in more detail.

The inclusion of semantic information in the exploration pipeline is used in [Kim et al., 2022] to reduce the likelihood of collision when using a 2D laser scanner for mapping with a ground robot. Object detections from an RGB camera are projected on the 2D cost map used for navigation to steer the robot away from obstacles undetectable by the laser scanner.

Preliminaries

Contents of Chapter

3.1	Notation	52
3.2	Pinhole Camera Model	52
3.3	Occupancy Mapping in supereight	53
3.4	Adaptive-resolution TSDF Mapping in supereight	55
3.5	Multi-resolution Occupancy Mapping in supereight2	57
3.6	OKVIS2 VI-SLAM System	58
3.7	Submapping using supereight2 and OKVIS2	59
	3.7.1 Reference Trajectory Anchoring	60
3.8	Path Planning Using Informed-RRT*	62

3.1 Notation

The following notation is used throughout this thesis.

$SE(3)$	The special Euclidean group of dimension 3; the group of 3D rigid transformations.
$SO(3)$	The special orthogonal group of dimension 3; the group of 3D rotations.
$\underline{\mathcal{F}}_W$	The world frame.
$\underline{\mathcal{F}}_B$	The MAV body frame.
$\underline{\mathcal{F}}_C$	The camera frame.
$a \in \mathbb{R}$	A lower-case symbol denotes a scalar.
$\mathbf{a} \in \mathbb{R}^m$	A bold lower-case symbol denotes an m -dimensional column vector.
$\mathbf{a} \in \mathbb{R}^m$	A bold italic lower-case symbol denotes an $m + 1$ -dimensional homogeneous column vector $\mathbf{a} = [\mathbf{a}^T \ 1]^T$.
$\mathbf{A} \in \mathbb{R}^{m \times n}$	A bold upper-case symbol denotes an $m \times n$ matrix.
${}_A \mathbf{a}$	A vector $\mathbf{a}$ expressed in coordinate frame A .
$T_{AB} \in SE(3)$	A rigid transformation from coordinate frame $\underline{\mathcal{F}}_B$ to coordinate frame $\underline{\mathcal{F}}_A$.

3.2 Pinhole Camera Model

We use a standard pinhole camera model, mapping a 3D point ${}_W \mathbf{p} = [x, y, z, 1]^T$ expressed in homogeneous coordinates in the world frame $\underline{\mathcal{F}}_W$ to the 2D pixel coordinates $\mathbf{u} = [u, v, 1]^T$. Given the homogeneous transformation T_{CW} from the world to the camera frame, the projection of the point onto the image plane is

$$\mathbf{u} = \mathbf{K} \pi \left(T_{CW} {}_W \mathbf{p} \right), \quad (3.1)$$

where $\pi(\cdot)$ is the perspective projection function

$$\pi({}_C\mathbf{p}) = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix}, \quad (3.2)$$

and $\mathbf{K}$ is the camera intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.3)$$

where f_u and f_v correspond to the camera's horizontal and vertical focal length in pixels and c_u and c_v to the horizontal and vertical offset to the optical axis also in pixels. We assume undistorted images are used and thus omit distortion in the pinhole camera projection model. The back-projection from 2D pixel coordinates $\mathbf{u}$ to a 3D point ${}_W\mathbf{p}$ in the world frame can be computed as

$${}_W\mathbf{p} = T_{WC} \pi^{-1}(\mathbf{K}^{-1}\mathbf{u}), \quad (3.4)$$

where $\pi^{-1}(\cdot)$ is the inverse perspective projection function

$$\pi^{-1}(\mathbf{u}) = \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix}. \quad (3.5)$$

For an image $\mathbf{C}$ with a pixel $\mathbf{u} = [u, v]^T$ we assume the pixel coordinate origin is at the top left corner, u increases rightwards and v increases downwards.

We assume the origin of the camera frame $\vec{\mathcal{F}}_C$ is located at the sensor's optical centre, its z axis is along the optical axis and its x axis is parallel to and has the same direction as the corresponding image's u axis.

3.3 Occupancy Mapping in *supereight*

We present a brief overview of the occupancy mapping pipeline of *supereight* as described in [Vespa et al., 2018].

Supereight is an efficient dense volumetric SLAM pipeline. Instead of a dense voxel grid, it uses an octree representation for the map. An octree is a hierarchical tree data structure used to partition 3D space. Each node in an octree represents

a volume which is subdivided into eight octants, each octant being a child of the node. This structure is repeated recursively until the desired leaf level, determining the spatial resolution of the octree. Using an octree for storing a dense volumetric 3D map allows scaling to large environments since only the observed regions need to be allocated. In *supereight*, octree leaves store $8 \times 8 \times 8$ blocks of voxels instead of individual voxels. This approach results in common operations accessing data stored close in memory, which improves performance by reducing CPU cache misses. Common operations include data update, interpolation, and raycasting, all of which are used extensively by the exploration methods presented in this thesis. A visualisation of the octree structure used in *supereight* is shown in Figure 3.1.

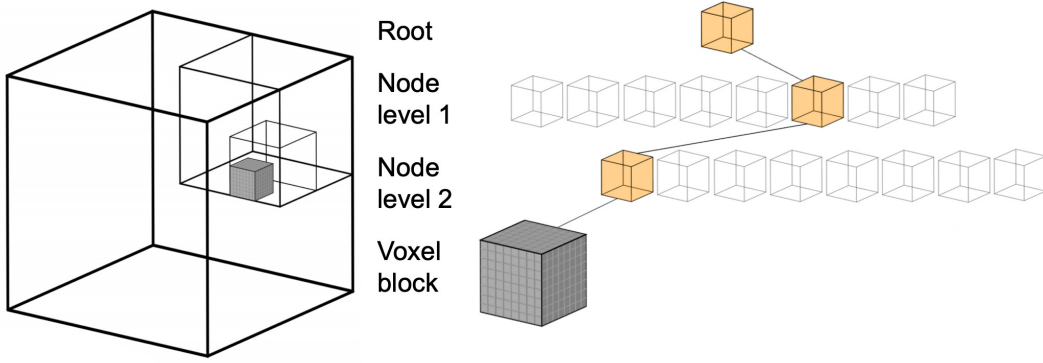


Figure 3.1: Visualisation of the *supereight* [Vespa et al., 2018] octree structure. The octree leaves store $8 \times 8 \times 8$ blocks of voxels instead of individual voxels. Image from [Funk, 2023].

The occupancy mapping pipeline of *supereight* explicitly stores both *free* and *unknown* space, allowing the use of the resulting map for safe path planning. Instead of storing occupancy probabilities directly, *supereight* occupancy mapping uses log-odds occupancy

$$l_o(\mathbf{v}) = \log_2 \frac{P_o(\mathbf{v})}{1 - P_o(\mathbf{v})}, \quad (3.6)$$

where $P_o(\mathbf{v}) \in [0, 1]$ is the occupancy probability and $l_o(\mathbf{v}) \in \mathbb{R}$ the log-odds occupancy of point $\mathbf{v}$. This results in

- *free* space having $P_o(\mathbf{v}) \in [0, 0.5) \Leftrightarrow l_o(\mathbf{v}) \in (-\infty, 0)$,
- *occupied* space having $P_o(\mathbf{v}) \in (0.5, 1] \Leftrightarrow l_o(\mathbf{v}) \in (0, \infty)$,
- and *unknown* space having $P_o(\mathbf{v}) = 0.5 \Leftrightarrow l_o(\mathbf{v}) = 0$.

Using log-odds allows occupancy updates to be additive instead of multiplicative, which can result in improved performance for some CPU architectures.

3.4 Adaptive-resolution TSDF Mapping in *supereight*

We present a brief overview of the adaptive-resolution TSDF mapping pipeline of *supereight* as described in [Vespa et al., 2019].

The measurement noise of typical depth sensors is proportional to the magnitude of the measurement, meaning higher measured depth values are more noisy. This can introduce artefacts in the surface reconstructions produced by 3D volumetric mapping pipelines that use a single voxel resolution, as is the case for [Vespa et al., 2018] described in Section 3.3. To alleviate this problem, an adaptive resolution update scheme for TSDF mapping was proposed in [Vespa et al., 2019].

The voxel block structure of [Vespa et al., 2018] is augmented with three mip-map levels called *scales*. *Scales* range from *scale* 3, aggregating an $8 \times 8 \times 8$ voxel volume, down to *scale* 1, aggregating a $2 \times 2 \times 2$ voxel volume, with *scale* 0 corresponding to the single-voxel data. The extra data stored in voxel blocks compared to [Vespa et al., 2018] results in a 14% increase in memory usage. The modified octree structure is depicted in Figure 3.2.

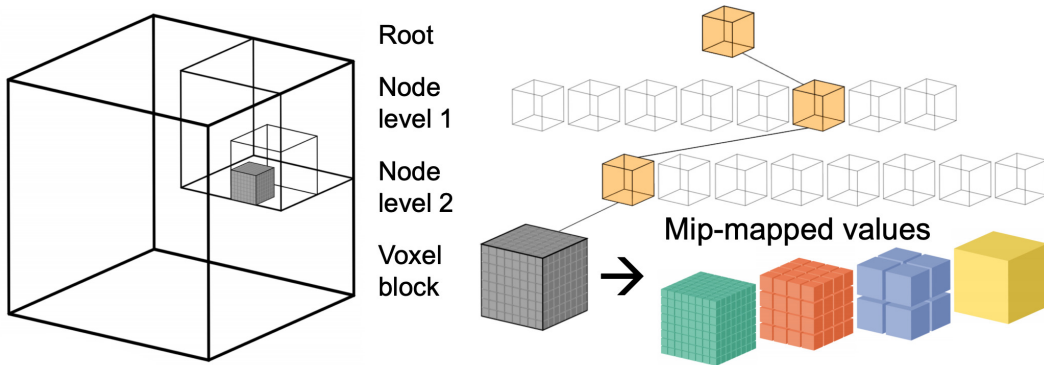


Figure 3.2: Visualisation of the *supereight* [Vespa et al., 2019] octree structure for adaptive-resolution TSDF mapping. The octree leaves store $8 \times 8 \times 8$ blocks of voxels and their mip-maps which aggregate multiple voxels into one. Image from [Funk, 2023].

For each depth image integration a single *scale* is assigned to each voxel block

3. Preliminaries

based on the distance of its centre from the camera and all the block’s data is updated at this *scale*. The *scale* is selected so that the area in pixels of the projection of a volume at this *scale* is as close as possible to the area of a pixel, as visualised in Figure 3.3. This reduces aliasing artefacts by ensuring the whole volume is updated using only a single depth measurement. The *scale* of a voxel block can change to a finer one, e.g. from *scale* 1 to 0, if it is observed from a smaller distance than earlier or to a coarser one if it is observed from a greater distance than earlier. A block’s *scale* may only change by one between subsequent depth image integrations and a hysteresis mechanism is used to avoid frequent scale changes. The TSDF field is updated using weighted mean fusion, as in [Newcombe et al., 2011] and [Vespa et al., 2018], with extra up- and down-propagation steps to keep the data consistent at all *scales*. In short, regions of the map close to the camera are updated at a fine spatial resolution while regions further away are updated at a coarser spatial resolution. Figure 3.4 shows 3D reconstructions of the same environment at increasingly coarse integration scales.

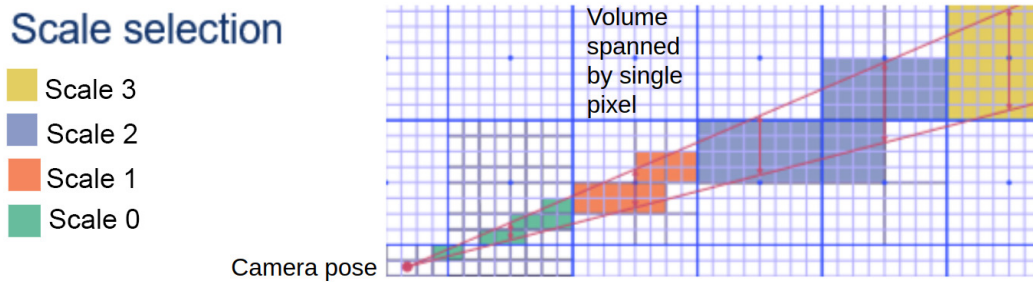


Figure 3.3: The integration *scale* selection for adaptive-resolution TSDF in *supereight* [Vespa et al., 2019]. The *scale* is selected so that the projected area of the volume to be updated is close to the area of a pixel. Image from [Funk, 2023].

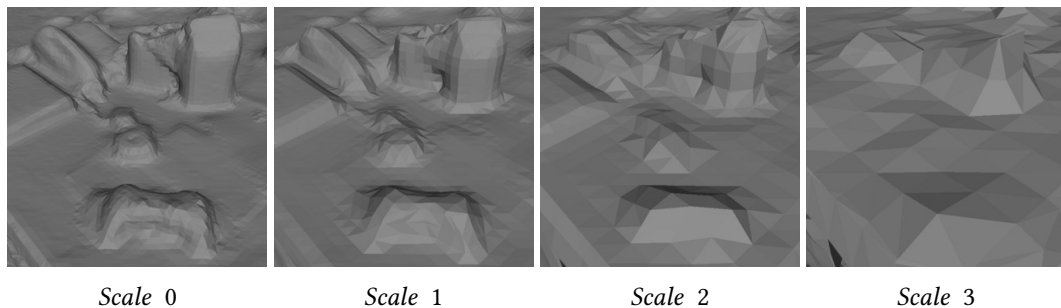


Figure 3.4: 3D reconstructions at increasingly coarse integration *scales* using the adaptive-resolution TSDF pipeline of *supereight* [Vespa et al., 2019].

3.5 Multi-resolution Occupancy Mapping in *supereight2*

We present a brief overview of the multi-resolution occupancy mapping pipeline of *supereight2* as described in [Funk et al., 2021].

A large amount of 3D volumetric mapping methods utilise an octree structure for fast and efficient map storage. However, they often do not take full advantage of its multi-resolution capabilities and offer either fast or accurate *free* space mapping but not both. For example, *OctoMap* [Hornung et al., 2013] updates free space at a fine resolution and then performs pruning, an approach that can provide accurate mapping but does not scale to large-scale, high-resolution maps. On the other hand, *supereight* [Vespa et al., 2018] updates *free* space at a constant, coarser resolution for efficiency but this approach can result in loss of information about *unknown* space, as shown in Figure 3.5 [Left].

To tackle these issues *supereight2* introduces a novel allocation and update strategy and a slightly modified octree structure compared to *supereight*. Compared to the octree used for adaptive-resolution TSDF mapping, as described in Section 3.4, data can be stored in all octree nodes instead of just in the leaves and only the finest *scale* needed is allocated, resulting in reduced memory usage.

In *supereight2* the multi-resolution capabilities of octrees are utilised by up-propagating the minimum and maximum log-odds occupancy values of the child nodes to the parent node. This allows for very efficient collision checking since whether a given region is safe or not can often be determined at a higher level of the octree without needing to traverse it down to the leaf level.

A novel map-to-image allocation process is employed by *supereight2* to ensure both efficient and accurate *free* space mapping by allocating and updating data as fine as necessary and as coarse as possible. First, a min-max pooling image pyramid is generated from the input depth image. Then each octree node in the camera frustum is projected onto the pooling image. The range of depth values it projects onto determines whether the node should be updated, skipped or subdivided and the same process recursively applied to its children. The update *scale* for voxel blocks (octree leaves) is decided in the same manner as in the adaptive-resolution TSDF case described in Section 3.4. This approach ensures the octree will be updated at the finest resolution required to represent the environment but no finer,

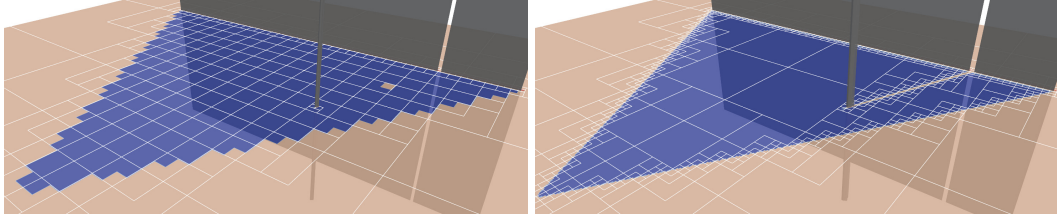


Figure 3.5: A semi-transparent slice of the log-odds occupancy through a *supereight* [Left] and a *supereight2* [Right] map. The 3D reconstruction mesh is shown in grey while *free* and *unknown* space is shown in blue and salmon respectively. The white lines denote the boundaries of allocated octants. It can be observed that in *supereight2* *free* space is allocated as coarsely as possible to conserve memory but also as finely as necessary to accurately represent the environment. Image from [Funk et al., 2021].

as shown in Figure 3.5 [Right].

3.6 OKVIS2 VI-SLAM System

OKVIS2 [Leutenegger, 2022] is a state-of-the-art, optimisation-based VI-SLAM system. It receives a set of two or more images, and IMU measurements and produces real-time estimates of the camera position, orientation and linear velocity with respect to the world frame $\mathcal{F}_w$ as well as gyro and accelerometer biases. [Leutenegger, 2022] can also produce state estimates at higher frequencies than the camera frame rate by using IMU pre-integration. This allows using it on more computationally constrained platforms, such as MAVs, where images have to be processed at a lower rate to maintain real-time operation.

The frontend initialises a new state for each image set and performs key-point matching, landmark triangulation, and place recognition. It marks the current image set as a keyframe if its covisibility with previous keyframes is below some threshold. Keyframe images are also passed through a segmentation CNN to filter-out keypoints on dynamic regions. The frontend is also responsible for re-localisation and loop closure initialisation.

The real-time estimator maintains and optimises the pose graph and fixes old states as new keyframes arrive. It also revives old observations as necessary to be used for loop closure optimisation. The full-graph estimator is executed asynchronously in the background once a loop closure is detected and the results are synchronised with the real-time graph after completion.

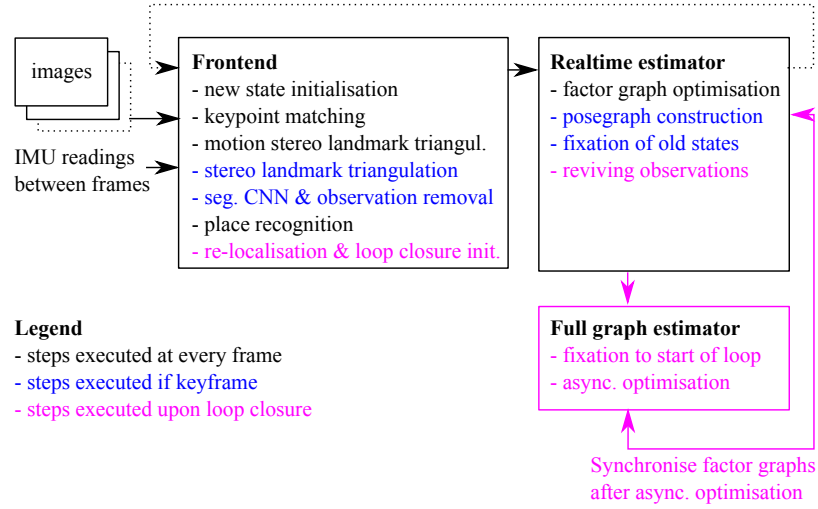


Figure 3.6: Overview of the *OKVIS2* VI-SLAM system. The frontend initialises a new state from an image set and corresponding IMU measurements, matches key-points, triangulates new landmarks, and attempts place recognition, re-localisation and loop closure detection. The real-time estimator maintains and optimises the pose graph and fixes old states. When loop closures are detected, an asynchronous full graph optimisation is performed and then synchronised with the real-time graph. Image from [Leutenegger, 2022].

3.7 Submapping using supereight2 and OKVIS2

Deployment of autonomous robots in large-scale environments requires the use of a SLAM system because robot state estimation systems are unavoidably affected by drift. Loop closures play an important role in such a setup by ensuring the estimated robot trajectory remains consistent when regions are revisited. However, the presence of loop closures makes using a single monolithic 3D volumetric map impractical, since making it consistent upon loop closures requires computationally expensive de-integration and re-integration of measurements, which in turns requires storing all past sensor measurements.

The most common solution to this problem is a submapping scheme, that is, a collection of smaller maps, each anchored in some way to the SLAM trajectory and transformed along with it. This is based on the assumption that SLAM drift is negligible locally and thus a small submap models its local region accurately enough. Submapping is highly efficient since only the transformations of the submaps with respect to the world frame $\mathcal{F}_W$ need to be updated on loop closures.

In Chapter 6 of this thesis we utilise a submapping approach based on *supereight2* and *OKVIS2*, as used in [Boche et al., 2024, Barbas Laina et al., 2024]. In this scheme each *supereight2* submap is anchored and centred on an *OKVIS2* keyframe state. Since new *OKVIS2* keyframes are generated once the covisibility of the current frame with previous keyframes is low enough, new keyframes are a good indicator that the robot has moved in the environment. As a result a new submap is created every n keyframes, with typical values $n \in [1, 10]$. Measurements are only integrated into the latest submap, which grows until a new submap is created and integration switches to the new one. In the current implementation there is no re-activation of old submaps or fusion of submaps upon loop closure, as was done in [Wang et al., 2021b]. As a result, once a new submap is created, all previous ones are considered finished and frozen. Figure 3.7 shows a visualisation of this submapping scheme.

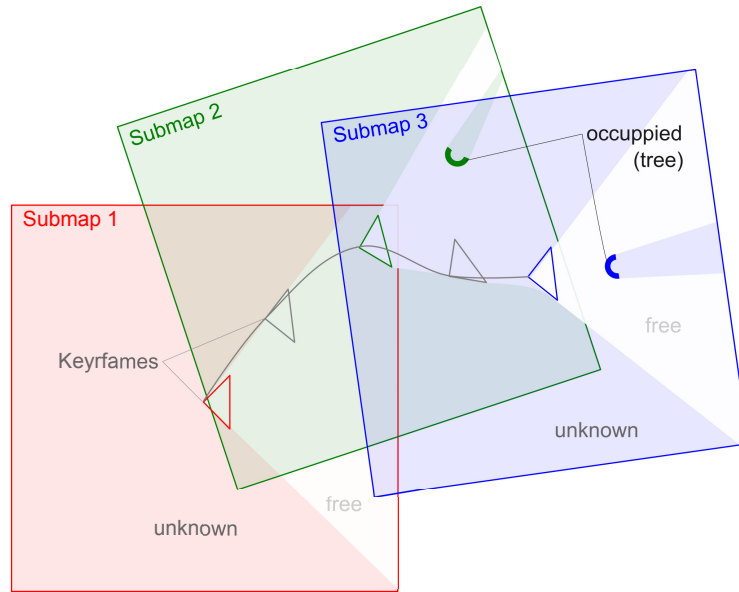


Figure 3.7: The submapping pipeline using *supereight2* and *OKVIS2*. Each *supereight2* submap is anchored on a keyframe and transformed along with it upon pose graph optimisation. Image from [Barbas Laina et al., 2024].

3.7.1 Reference Trajectory Anchoring

The state estimates from a VI-SLAM system can have significant discontinuities over time, for example due to a large loop closure. This poses a safety risk if these

state estimates are used by an MAV controller without adjusting the current reference appropriately. If the reference trajectory is not adjusted, it may result in a collision since it now corresponds to a different trajectory relative to the MAV. Moreover, a discontinuity in the MAV state estimate without appropriate adjustment of the reference trajectory will result in a discontinuity in the reference error, which can cause the MAV to fly aggressively.

A solution for these issues has been proposed in [Barbas Laina et al., 2024] in the form of reference trajectory anchoring. Once a new MAV reference trajectory is produced, for example by the exploration planner, each trajectory state is anchored relatively to a number of keyframe states. Upon update of the anchor states, each trajectory state is recomputed using a weighted average based on its anchors. A graphical overview of the approach is shown in Figure 3.8.

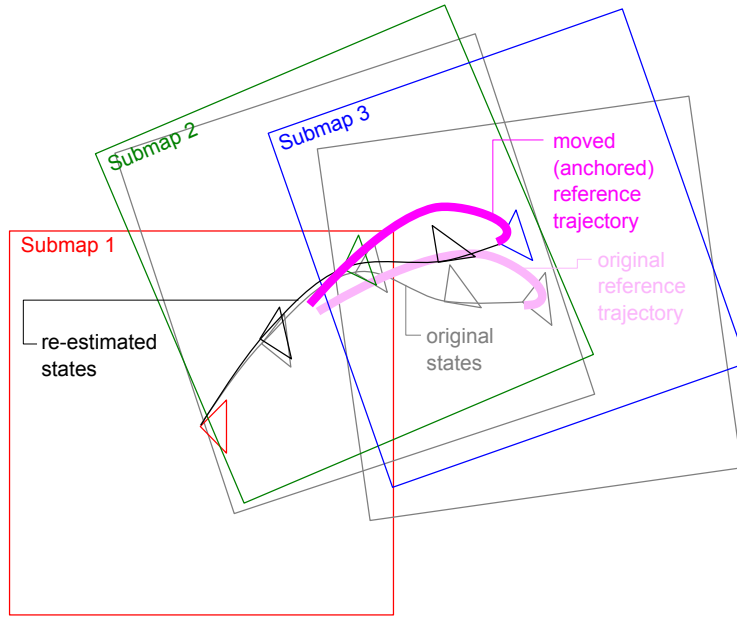


Figure 3.8: Reference trajectory anchoring due to an estimated state discontinuity, e.g. due to loop closure, in the scenario depicted in Figure 3.7. As the submaps move along with the keyframe states they are anchored on, the MAV reference trajectory anchored on multiple keyframe states is also deformed accordingly. This scheme avoids large discontinuities in the MAV trajectory tracking error, preventing aggressive and unsafe behaviour. Image from [Barbas Laina et al., 2024].

An MAV state $\mathbf{x}$ estimated by *OKVIS2* consists of the position ${}_w\mathbf{r}_B$, orientation $\mathbf{q}_{wB}$ and linear velocity ${}_w\mathbf{v}$ of the MAV body frame $\underline{\mathcal{F}}_B$ with respect to the world

frame $\xrightarrow{w}$:

$$\mathbf{x} = ({}_w\mathbf{r}_B, \mathbf{q}_{wB}, {}_w\mathbf{v}) \in \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3. \quad (3.7)$$

A reference trajectory consists of a series of timestamped MAV states:

$$\mathbf{x}_i^{\text{ref}} = ({}_w\mathbf{r}_{B_i}^{\text{ref}}, \mathbf{q}_{wB_i}^{\text{ref}}, {}_w\mathbf{v}_i^{\text{ref}}), \quad i \in \{1, \dots, I\}. \quad (3.8)$$

Each reference state $\mathbf{x}_i^{\text{ref}}$ is anchored to the S nearest *OKVIS2* states, forming a set $\mathcal{S}_i$ of anchor states $\mathbf{x}_s$, $s \in \{1, \dots, S\}$. The S nearest states are computed using the distance metric:

$$d_{i,s} = \| {}_w\mathbf{r}_{B_i}^{\text{ref}} - {}_w\mathbf{r}_{B_s} \| . \quad (3.9)$$

Every pair i, s is associated with the relative transformation $T_{B_s B_i}$ and a weight $w_{i,s}$:

$$w_{i,s} = \frac{\frac{1}{d_{i,s}}}{\sum_{i=1}^S \frac{1}{d_{i,s}}}. \quad (3.10)$$

After each MAV state update, reference trajectory positions are updated as follows:

$${}_w\mathbf{r}_{B_i}^{\text{ref}'} = \sum_{s=1}^S w_{i,s} T_{wB_s B_i} \mathbf{r}_{B_i}, \quad i \in \{1, \dots, I\}, \quad (3.11)$$

where T_{wB_s} is the latest pose estimate of *OKVIS2* for anchor s . The orientation $\mathbf{q}_{wB_i}^{\text{ref}'}$ is updated to the weighted average of $\mathbf{q}_{B_s B_i}$, $s \in \{1 \dots S\}$, using the weights from Equation (3.10), using the method from [Markley et al., 2007]. The resulting orientation changes are applied to the reference velocities in order to keep them consistent.

3.8 Path Planning Using Informed-RRT*

An integral part of an autonomous MAV system is safe path planning. Deploying an MAV in an unknown environment lacking prior infrastructure requires it to be capable of planning paths using just the environment maps created using its on-board sensors and state estimates.

RRTs, introduced by [LaValle, 1998], are one of the most popular class of algorithms for efficient path planning. The goal is to grow a tree structure filling the admissible, that is, safe, regions of the search space. Then a tree traversal can be used to find the best path on the tree from the start to the goal state. The tree is initialised to the starting state and is incrementally grown until the goal state is

connected to the tree. Growing happens by randomly sampling a new state and attempting to connect it to the nearest state currently in the tree. If the connection is feasible, meaning it passes entirely through safe space, the sampled state is added to the tree. Sometimes, in order to limit the growth of the tree, a state some distance along the connection is added to the tree instead of the sampled state. One drawback of the original RRT algorithm is that it results in sub-optimal paths. As a result several extensions that attempt to quickly converge to an optimal solution have been proposed.

The RRT* algorithm introduced in [Karaman and Frazzoli, 2011] is an asymptotically optimal version of RRT by rewiring the tree so that the parent of each node is along the best path to the start. A further improvement over RRT* is Informed RRT*, proposed in [Gammell et al., 2014]. It achieves much higher sample efficiency compared to plain RRT* by limiting the search space. Once an initial solution is found, Informed RRT* samples states only inside a prolate hyperspheroid enclosing the start and goal states instead of sampling the whole search space. As better solutions are found, the volume of the prolate hyperspheroid decreases, further reducing the search space. This allows finding equivalent paths to plain RRT* in a fraction of the time. A comparison of the trees spanned by RRT* and Informed RRT* is shown in Figure 3.9. In this thesis we use the Informed RRT* implementation from the Open Motion Planning Library (OMPL) [Sucan et al., 2012].

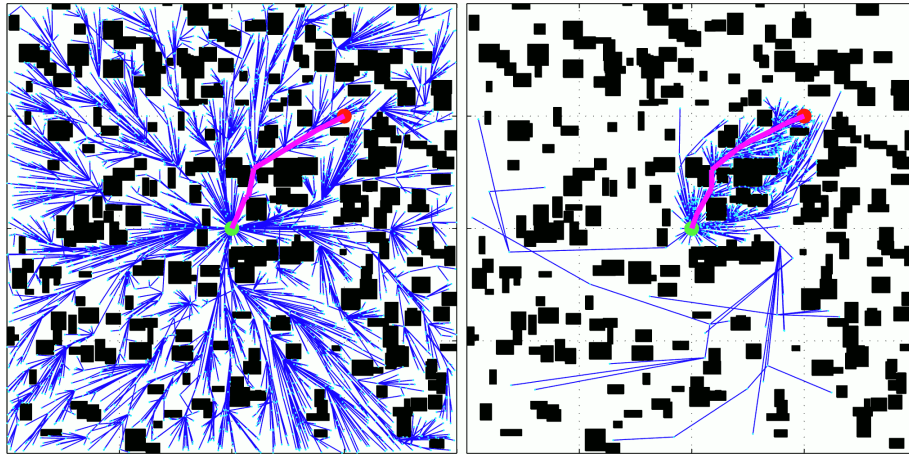


Figure 3.9: Trees spanned by RRT* [Left] and Informed RRT* [Right] in a 2D environment. Obstacles are shown in black, trees in blue, the best paths in pink, and the start and goal states in green and red respectively. While the resulting paths are practically the same, sampling in Informed RRT* was concentrated in a small subset of the search space. Image from [Gammell et al., 2014].

Information-Driven Autonomous Exploration

Contents of Chapter

4.1	Introduction	65
4.2	Related Work	66
4.3	Problem Formulation	67
4.3.1	Environment Model	68
4.3.2	MAV Model	68
4.4	Methodology	69
4.4.1	Map Representation	70
4.4.2	Frontier Detection	71
4.4.3	Candidate Position Sampling	71
4.4.4	Path Planning to Candidate Positions	72
4.4.5	Yaw Optimisation and Candidate Pose Evaluation . .	72
4.5	Evaluation	75
4.5.1	Apartment Environment Simulation	76
4.5.2	Maze Environment Simulation	77
4.5.3	Powerplant Environment Simulation	78
4.5.4	Computational Complexity	79
4.5.5	Real World Experiments	80
4.6	Conclusion	81

Parts of this chapter appear in:

Anna Dai, Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas and Stefan Leutenegger (2020). **Fast Frontier-Based Information-Driven Autonomous Exploration with an MAV**. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Dai et al., 2020]

4.1 Introduction

The use of MAVs in applications such as mining, outdoor large scale inspection, precision agriculture and search and rescue have gained much popularity recently. Fast and thorough autonomous exploration is a key capability required for safe and effective operation. Due to their agility and speed, MAVs are well suited for mapping 3D environments. As the exploration algorithm must run on-board an MAV with limited computational resources and flight time, it should be fast and lightweight. The most common exploration planning strategies are frontier-based and sampling-based exploration which are compared in [Juliá et al., 2012]. Frontier-based methods generate their planning tasks from frontiers which are defined as boundaries between free and known space, whereas sampling-based methods most commonly grow RRTs [LaValle, 1998] to compute an exploration path.

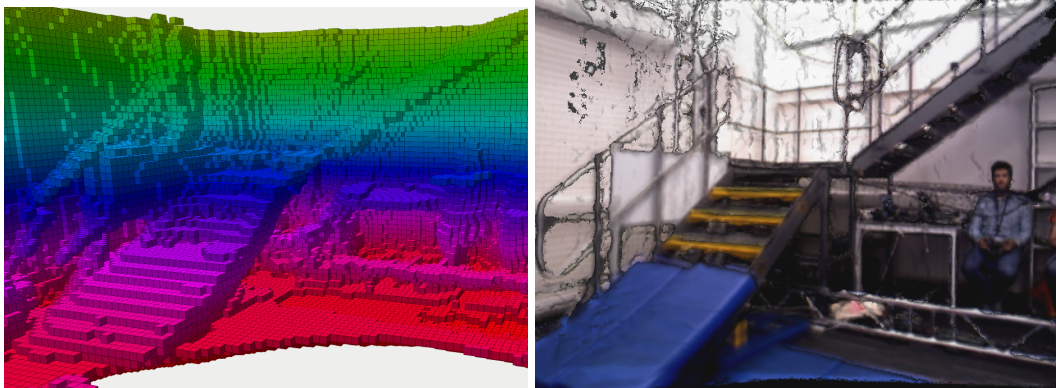


Figure 4.1: In this chapter, a light-weight exploration algorithm for MAVs is presented. [Left] The map created on-board the MAV in real-time using a voxel resolution of 10 cm which is used for exploration planning and collision avoidance. [Right] A high resolution render of a map created offline from the depth and colour images collected in the experiment. The map was created using the pipeline presented in [Vespa et al., 2019].

In this chapter, we present an exploration strategy which can achieve real-time performance on-board an MAV. The main contributions of the strategy are as follows:

- Frontier-based exploration is combined with sampling-based exploration by sampling candidate next-views from the map frontiers. This approach results in exploration focused near frontiers, like frontier-based methods do. At the same time it requires considerably fewer candidate next-views than typical sampling-based methods due to sampling being constrained to frontier regions.
- By leveraging the implicit voxel grouping in the octree map representation, the computationally expensive step of frontier voxel clustering in classical frontier-based methods can be avoided. This is achieved by considering frontier voxels as clusters if they reside in the same octant of the octree.
- Performing a computationally expensive expected sensor measurement map update like conventional information-theoretic methods is avoided by observing that only relative ranking between candidate next-views is needed. Thus a much faster map entropy estimation using sparse raycasting is performed.

4.2 Related Work

A considerable amount of work has been carried out on autonomous exploration and multiple kinds of approaches have been proposed.

In the original work on frontier exploration [Yamauchi, 1997], the closest frontier from the current robot position is chosen. In [Cieslewski et al., 2017], priority is given to maintaining a high MAV velocity during the exploration task. The method introduced in [Gao et al., 2018b] focuses on keeping track of visited key nodes in a topological graph while conducting frontier exploration. A more recent work on frontier-based autonomous inspection is presented in [Faria et al., 2019]. Frontiers can be used to provide global exploration guidance

The core idea of sampling-based exploration approaches is to sample potential poses which could contribute to mapping the unknown volume. This strategy avoids the map-wide computation of frontiers but requires the evaluation of each

sample. *RH-NBVP* [Bircher et al., 2016] has shown that the Next-Best-View (NBV) method [Connolly, 1985] can be applied to 3D exploration. The NBV is found by growing an RRT to sample positions and yaw angles in free space and evaluating them based on the observable volume of unknown space. This method is improved in [Witting et al., 2018] where a history of visited locations is maintained to avoid being stuck at local minima. *AEP* [Selin et al., 2019] further improves the sampling efficiency by sampling only 3D positions and computing the yaw angle that results in the highest expected gain. It also provides global guidance by modelling the gain over the environment using a Gaussian process. However the number of samples used by the Gaussian process keeps increasing as the exploration progresses, limiting its utility to larger environments. Various approaches for computing the volumetric gain of candidate views are compared in [Delmerico et al., 2018]. A frontier-oriented volumetric hierarchical ray tracing method is proposed in [Bissmarck et al., 2015] and benchmarked against [Vasquez-Gomez et al., 2013] in computation time, mapping efficiency and estimation error. *RH-NBVP* is extended in [Papachristos et al., 2017] to consider localisation uncertainty as a secondary sub-task. A similar framework is used in [Dang et al., 2018b] to guide the MAV towards visually salient regions.

Other methods guide the exploration so that map entropy is reduced [Charrow et al., 2015b]. These exploration strategies most often use the Shannon entropy [Scott and Jerath, 2018, Palazzolo and Stachniss, 2018]. In [Carrillo et al., 2017] the combination of Shannon and Rényi entropy results a utility function which balances between robot localisation and map uncertainty. A common issue with entropy-based methods is their high computational requirements. In [Kaufman et al., 2018] mapping is performed in 3D while motion planning is done in 2D to reduce the computational complexity, under the assumption that human-made structures are mostly uniform in the vertical axis.

4.3 Problem Formulation

The goal of this work is to use a sensor-equipped MAV to explore and map an unknown space. Thus, the MAV must be able to map its environment and plan safe exploration paths.

4.3.1 Environment Model

Let $V \subset \mathbb{R}^3$ be a bounded volume to be explored. In most non-trivial environments, there are regions $V_{\text{unob}} \subset V$ that can not be observed by the sensor, such as the interior of solid objects or passages too small for the robotic platform to explore, limiting exploration to the observable subset $V_{\text{obs}} = V \setminus V_{\text{unob}}$. Thus autonomous exploration creates a map $M : V_{\text{obs}} \rightarrow [0, 1]$, a field of occupancy probabilities over V_{obs} , with the goal of minimising the field entropy. This means updating M to reflect which regions of V_{obs} are *free* and which ones are *occupied*. Another concept useful in exploration is that of frontiers. Frontiers are the boundaries between *free* and *unknown* space. They mark observable regions that can lead to an expansion of the map if observed. The exploration termination condition can be equivalently expressed in terms of frontiers as the state where there are no more frontiers left.

Initially, $M(\mathbf{v}) = 0.5, \forall \mathbf{v} \in V_{\text{obs}}$ if no prior information is available and the robotic platform has to map and navigate the unknown space. Path planning, collision avoidance, and exploration planning have to be performed online as there is no *a priori* knowledge of the environment. Thus it is important for the whole exploration pipeline to run efficiently on-board the robotic platform.

4.3.2 MAV Model

Our exploration method is specifically tailored for MAVs. It is assumed that the MAV's state $\mathbf{x}$ consists of its position $\mathbf{p} = [x, y, z]^T \in V$ and yaw angle, $\psi \in [0, 2\pi)$, relative to the world frame $\underline{\mathcal{F}}_W$ thus $\mathbf{x} = [x, y, z, \psi]^T \in V \times [0, 2\pi)$. This is the 4-DoF subset of the MAV's configuration space that is controllable while hovering, as described in Section 2.3. It is also assumed that the MAV has a maximum linear velocity $v_{\text{max}} \in \mathbb{R}^+$ and maximum yaw rate $\omega_{\text{max}} \in \mathbb{R}^+$. The MAV is also equipped with a depth camera mounted on a fixed position, with maximum sensor range $d_{\text{max}} \in \mathbb{R}^+$ and horizontal and vertical field of view α_h and α_v respectively. For collision checking, we assume the MAV is contained inside a sphere with centre $\mathbf{p}$ and safety radius $R \in \mathbb{R}^+$.

4.4 Methodology

We propose a frontier sampling, information gain-based exploration approach. It takes inspiration from both frontier-based and sampling-based exploration methods. Frontiers are good indicators of the regions the exploration should be focused on, which are ignored by conventional sampling-based exploration algorithms. By sampling candidate next-views at the frontiers there is no need for clustering of the individual voxels into larger frontiers, thus removing the associated large computational cost. A utility function expressing expected information gain over time is used to evaluate the candidate next-views. This is a more meaningful metric than just counting unobserved voxels and does not require a tuning parameter to take the path cost into account.

An occupancy mapping server [Vespa et al., 2018] is kept running in the background, which continuously integrates depth image and pose pairs into the map and extracts the frontiers. At the beginning of the exploration and after some measurements have been integrated into the map, the exploration planner is called. At each planning iteration it executes the following steps in sequence:

1. A pre-defined number of candidate goal positions are sampled from frontier voxels (Section 4.4.3).
2. Collision-free paths from the current position to each candidate position are planned (Section 4.4.4).
3. A yaw angle is associated to each candidate position, creating a candidate pose, and each pose is evaluated based on a utility function expressing information gain over time (Section 4.4.5).
4. The candidate pose with the highest utility is selected as the next goal.

The map is continuously updated as the MAV moves towards the goal. Once the goal is reached, the next planning iteration begins. The exploration is considered complete when there are no more frontier clusters remaining. A graphical overview of the proposed approach is presented in Figure 4.2.

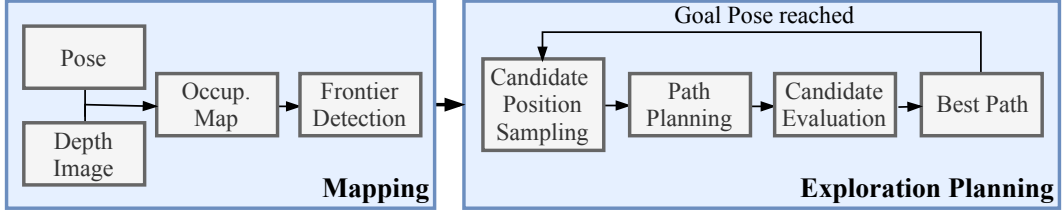


Figure 4.2: The mapping system runs continuously in the background while the exploration planner produces a new path each time the previous path is completed.

4.4.1 Map Representation

We chose to use an occupancy map for three main reasons. First, it provides an explicit representation of free space in which collision-free paths can be easily planned. Second, an occupancy map is also inherently suitable for integrating noisy sensor measurements. Finally, it provides an implicit measure of map quality through its entropy.

The occupancy mapping framework used in this work is *supereight* [Vespa et al., 2018]. *Supereight* uses an octree structure to store the map. An octree is a tree data structure each node of which subdivides the 3D space it corresponds to into 8 octants which are represented as its children nodes in the octree. *Supereight* also provides efficient map updates and queries by using Morton codes for spatial indexing. Instead of storing single voxels at the octree leaf level, *supereight* stores blocks of voxels at the leaf level, typically $8 \times 8 \times 8$ voxels in size. The side length in metres of a single voxel is called map resolution and is denoted r . The map is continuously updated as new pose and depth image pairs arrive. The system does not assume any particular pose tracking method. Thus any system can be run in parallel to the exploration and provide an estimate of the MAV’s pose, e.g. a visual-inertial odometry pipeline or a motion capture system.

We extend *supereight* to store whether each voxel is a frontier and perform frontier detection, as described in the following section. Additionally, occupancy probabilities are up-propagated through the octree levels to facilitate efficient collision queries: after a new depth measurement is integrated into the map, each parent node whose children were updated will also have its occupancy probability updated. Specifically, the occupancy probability of the parent is set to the maximum of the occupancy probabilities of its children. This way, if an octree node has an occupancy probability smaller than 0.5, it is guaranteed that its children also

have an occupancy probability smaller than 0.5.

4.4.2 Frontier Detection

The map frontiers are detected during the map update process. The frontiers are stored as a sorted list F containing Morton codes corresponding to voxel blocks (octree leaves) with one or more frontier voxels. An implicit frontier clustering at the voxel block level is achieved by considering voxels in the same voxel block as a frontier cluster. Thus any computationally intensive clustering methods as in [Charrow et al., 2015b] are avoided.

At each depth measurement integration into the map, the voxels inside the camera frustum are updated. Each of the updated voxels is tested for being a frontier voxel. Voxels are considered frontiers if their occupancy probability is lower than 0.5, while one or more of their 6 face neighbour voxels has an occupancy probability of exactly 0.5. More intuitively, frontier voxels are free voxels located next to completely unobserved voxels. The Morton code list is updated as new frontier voxels emerge and previously unobserved regions are observed. The latter leads to removal of frontier voxels. By only considering the last updated voxels for the map frontiers update, a map-wide operation is avoided. This frontier update is performed continuously at each sensor measurement integration.

4.4.3 Candidate Position Sampling

At the beginning of each planning iteration, a predefined number N_c of candidate positions $\mathbf{p}_i = [x_i, y_i, z_i]^T \in \mathbb{R}^3$, $i \in \{1 \dots N_c\}$ is uniformly sampled from the frontier voxel blocks in F . Initially, the number of frontier voxels in each frontier voxel block is obtained. Frontier voxel blocks containing fewer frontier voxels given a certain threshold are ignored for the duration of the candidate position sampling, resulting in the filtered frontier list $\tilde{F}$. Since $\tilde{F}$ is sorted and due to the spatial indexing of Morton codes, a more spatially uniform candidate position sampling can be achieved by sampling every $\lceil N_{rem}/N_c \rceil$ th frontier voxel block in $\tilde{F}$, where N_{rem} is the number of frontier voxel blocks in $\tilde{F}$ and $\lceil \cdot \rceil$ denotes the ceiling of the argument. For each of the selected frontier voxel blocks, one of its frontier voxels is randomly selected and its coordinates used as a candidate position $\mathbf{p}_i$. The current

MAV position is always added to the candidates to ensure that a rotation around the yaw axis without any movement is also considered as candidate pose.

4.4.4 Path Planning to Candidate Positions

The Open Motion Planning Library (OMPL) [Sucan et al., 2012] is used for computing paths to the candidate positions. The integrated planning algorithm is the informed RRT* [Gammell et al., 2014] with path simplification. The octree map structure is exploited to perform efficient collision checking. Free space queries are made by checking the occupancy probability against a threshold and are first performed at a higher octree level, before checking at the single-voxel level. A sphere around the MAV is used for collision checking for points and a cylinder for line segments. The path is computed in $\mathbb{R}^3$ and the corresponding yaw angles are computed at a later stage.

Since candidate positions are sampled at the frontier, hence close to unknown space, the MAV cannot safely reach them due to its finite dimensions. In this case, OMPL computes a collision-free path which ends before the candidate position, typically at a distance near the MAV's safety radius. The candidate position is then updated to this safe path endpoint resulting in the path $W_i(\mathbf{p}, \mathbf{p}_i)$ from the current MAV position $\mathbf{p}$ to the updated candidate. The path consists of one or more line segments.

In receding horizon exploration methods [Bircher et al., 2018, Selin et al., 2019], the MAV moves only to the first point of a planned path before performing another planning iteration. This sometimes has the effect that the MAV moves back and forth in a small area since at subsequent planning iterations it may move to the first point of paths to different endpoints (i.e. from different frontiers). Thus the MAV is temporarily stuck in a small region. In our approach, the MAV follows the path to its endpoint, committing to exploring a single frontier before moving towards another goal.

4.4.5 Yaw Optimisation and Candidate Pose Evaluation

The map entropy is computed at each candidate position by performing a 360° sparse raycasting. Each candidate position $\mathbf{p}_i$ is converted into a candidate pose $\mathbf{x}_i = [x_i, y_i, z_i, \psi_i]^T$ by optimising the yaw angle for the highest entropy given the

360° raycast. The estimated travel time to the candidate pose is computed based on the MAV's maximum linear velocity $v_{\max}$ and maximum yaw rate $\omega_{\max}$. The utility function of a candidate goal pose is computed as the map entropy over the estimated path travel time. Finally the candidate pose with the highest utility is selected as the next goal pose. We will detail these steps in the following.

Sparse Raycasting and Yaw Optimisation

Instead of random sampling the MAV yaw angle using the informed RRT*, it is optimised by performing a sparse 360° raycasting which at the same time computes the map entropy along each ray. It has been shown [Oleynikova et al., 2018a] that sparse raycasting can effectively approximate a full raycasting at a fraction of the computational cost.

Given the sensor's vertical field of view α_v and its maximum range $d_{\max}$, the raycasting is performed with a horizontal angle increment $\delta\psi$ for the yaw angle and a vertical angle increment $\delta\theta$ for α_v , with the rays being stopped at a distance $d_{\max}$ from $\mathbf{p}_i$. The entropy for a single voxel $\mathbf{v}$ is calculated using Shannon's information theory

$$H(\mathbf{v}) = -P_o(\mathbf{v}) \ln P_o(\mathbf{v}) - (1 - P_o(\mathbf{v})) \ln (1 - P_o(\mathbf{v})), \quad (4.1)$$

where $H(\mathbf{v})$ denotes the map entropy and $P_o(\mathbf{v})$ the occupancy probability of a voxel $\mathbf{v}$.

The entropy along a ray is computed as the sum of the entropy of the voxels along the ray until the first voxel $\mathbf{v}$ with $P_o(\mathbf{v}) > 0.5$ or $d_{\max}$ is reached, whichever comes first. By taking into account the entropy of both observed and unobserved voxels, we evaluate both unknown regions and regions where the map is still uncertain and of low quality. Thus it acts as a measure of both map coverage and quality. The raycasting is performed on the current map and computes a 360° estimated entropy map, with each pixel corresponding to a single ray, like the one shown in Figure 4.3 [Bottom]. The brighter a pixel is, the higher the entropy for the corresponding ray. The corresponding sparse depth map is shown in Figure 4.3 [Top]. Lighter shades of grey denote rays that hit an occupied voxel closer to the candidate position, while black denotes rays that did not hit any occupied voxels up to a distance $d_{\max}$, producing an invalid depth measurement. This can be either due to free space extending to a distance greater than $d_{\max}$ or due to the existence of

voxels with high map entropy, as is the case for the rightmost and leftmost invalid depth regions respectively in Figure 4.3 [Top].

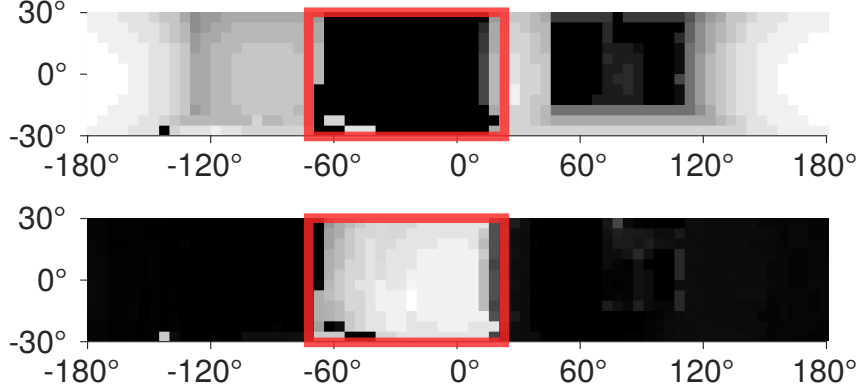


Figure 4.3: Sparse raycasted depth map [Top] and corresponding entropy map [Bottom]. The boundaries of the camera frustum at the yaw angle that results in the highest map entropy are shown in red.

To obtain the optimal yaw for the candidate position, a sliding window summation with an angular width α_h is performed on the entropy map. For each yaw value, the entropy sum for the rays that fall inside the sensor’s field of view is computed. The yaw angle ψ_i with the highest cumulative map entropy is selected, resulting in the candidate pose $\mathbf{x}_i$. We use $H(\mathbf{x}_i)$ to denote the map entropy obtained by raycasting from candidate pose $\mathbf{x}_i$. The boundaries of the camera frustum at the yaw angle that results in the highest map entropy are shown in red in Figure 4.3.

Utility Function

The utility for a candidate pose $\mathbf{x}_i$ given the path W_i to it is the entropy of the portion of the map visible from the pose over the time it takes to complete the path W_i . This results in a balance between exploring the MAV’s immediate surroundings by penalising time-consuming paths and moving towards more distant frontiers leading to potentially completely unexplored regions by favouring high map entropy. By formulating the utility as a ratio, this balance is achieved without the need for any tuning parameter.

The estimated information gain computation using sparse raycasting was described in the previous section. The estimated time $T(W_i)$ to complete path W_i is computed by assuming the MAV always flies at its maximum linear speed and rotates at the maximum rate when needed. Thus the total path travel time $T(W_i)$

is estimated as the maximum between the time the MAV needs to move from its start point $\mathbf{p}$ to its final point $\mathbf{p}_i$ at $v_{\max}$ and the time it takes to rotate from the start yaw ψ to its final yaw ψ_i at $\omega_{\max}$. Although this estimate is not accurate due to the constraints imposed by the MAV's dynamics, it is a suitable approximation. The utility function serves to provide a ranking of the candidate poses.

Thus the utility of a candidate pose is computed as

$$u(\mathbf{x}_i, \hat{W}_i) = \frac{H(\mathbf{x}_i)}{T(\hat{W}_i)}, \quad (4.2)$$

and the candidate pose $\hat{\mathbf{x}}_i = [\hat{\mathbf{p}}_i^T \hat{\psi}_i]^T$ with the highest utility is selected as the next goal pose.

Since the path $W_i(\mathbf{p}, \hat{\mathbf{p}}_i)$ consists of a number of points with no associated yaw, each point in the path has to be matched to a yaw angle before it is supplied to the MAV's controller. The initial and final points of the path already have associated yaw angles ψ and $\hat{\psi}_i$ from the current $\mathbf{x}$ and best candidate pose $\hat{\mathbf{x}}_i$ respectively. For each intermediate path point a yaw angle is assigned by performing the same yaw optimisation as for $W_i(\mathbf{p}, \mathbf{p}_i)$, resulting in the final path $\hat{W}_i(\mathbf{x}, \hat{\mathbf{x}}_i)$ to the goal pose. Essentially, for each intermediate point of the path to the best candidate pose a sparse raycasting and related map entropy computation is performed. The number of intermediate points depends on the geometry of the space but due to the OMPL path simplification it is in general low. Since this intermediate yaw optimisation is performed only for the path to the best candidate, its effect on the total computation time is limited.

4.5 Evaluation

The exploration algorithm has been evaluated in both simulated and real world experiments. All simulated experiments were performed in Ubuntu 18.04 using ROS Melodic [Open Robotics, 2018b] and the RotorS simulator [Furrer et al., 2016]. The MAV model used in the simulation is the Ascending Technologies FireFly hexacopter equipped with a VI-Sensor mounted at a 15° angle. As a baseline we use *RH-NBVP* [Bircher et al., 2016], a receding-horizon MAV exploration planner that iteratively samples candidate next views and moves towards the one with the highest utility. *RH-NBVP* was selected because it is a widely used baseline and because at a high level it operates similarly to our proposed approach. A more detailed

overview of *RH-NBVP* is available in Section 2.5.3. All simulations were run on an Intel Core i7-8750H CPU operating at 2.20 GHz and compiled with g++ 7.4.0 using the O3 optimisation level. The parameters used for both the simulated and real-world experiments can be found in Table 4.1.

Parameter		Simulation	Real world
Voxel resolution	r	0.1 m	0.1 m
MAV radius	R	0.5 m	0.8 m
MAV maximum linear velocity	$v_{\max}$	1.5 m/s	0.1 m/s
MAV maximum angular velocity	$\omega_{\max}$	0.75 rad/s	0.15 rad/s
Maximum depth measurement	$d_{\max}$	5 m	4 m
Horizontal/vertical sensor FoV	$[\alpha_h, \alpha_v]$	$[115^\circ, 60^\circ]$	$[58^\circ, 45^\circ]$
Candidate poses sampled	N_c	20	20

Table 4.1: Experiment parameters.

4.5.1 Apartment Environment Simulation

Several simulations were conducted in the $10 \times 20 \times 3$ m *Apartment* environment used in [Bircher et al., 2018, Selin et al., 2019]. Our algorithm was compared with *RH-NBVP* [Bircher et al., 2018]. Figure 4.4 shows the explored volume over time, averaged over 10 runs, for both algorithms using a voxel resolution of $r = 0.1$ m and $r = 0.4$ m and a horizontal field of view of $\alpha_h = 90^\circ$. It can be observed that our method explores the whole environment substantially faster, especially when using a high resolution map. An interesting property of our method is that it does not require any arbitrary initialisation movement, like the 360° in-place rotation in [Bircher et al., 2018]. Instead, the utility function usually gives higher utility to candidate poses that result in the MAV performing a similar rotation in the beginning of the exploration. Our method explores 95% of the environment in 80 s and 151 s for a resolution of 0.4 m and 0.1 m respectively. These are lower than the respective times reported in [Selin et al., 2019], although it should be noted that we were unable to run their planner on our hardware, despite our best efforts, so the results might not be directly comparable, especially since the hardware used by the authors is not stated.

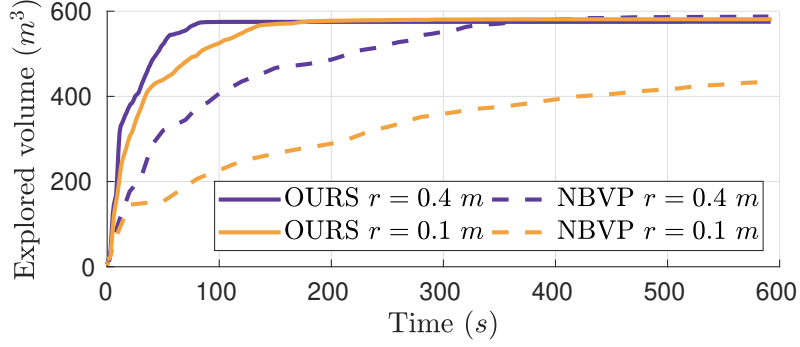


Figure 4.4: Explored volume over time in the *Apartment* environment.

4.5.2 Maze Environment Simulation

Simulations were also conducted in the more complex $20 \times 20 \times 2.5$ m *Maze* environment from [Oleynikova et al., 2017]. Figure 4.5 shows the explored volume over time, averaged over 10 runs, for both our method and *RH-NBVP* using a voxel resolution of $r = 0.1$ m and $r = 0.2$ m. The planners were not run at a resolution of 0.4 m because this environment contains some narrow corridors that the MAV cannot navigate through at such a coarse resolution. Our method performs significantly faster than *RH-NBVP* in this case since the random sampling of *RH-NBVP* does not always detect unexplored regions quickly. This results in the MAV being stuck in a portion of the map for a long time before moving towards unexplored regions. Our planner does not exhibit this behaviour due to its use of frontiers to reliably guide the MAV towards unexplored space. Our method explores 95% of the environment in 177 s and 330 s for a resolution of 0.2 m and 0.1 m respectively. Figure 4.6 shows a top-down view of a map of the *Maze* environment created by our algorithm.

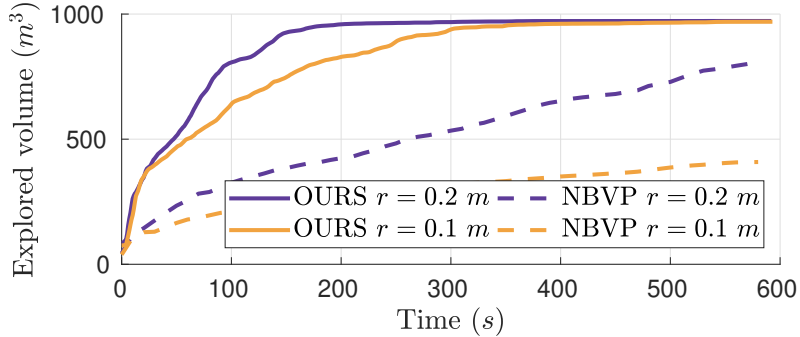


Figure 4.5: Explored volume over time in the *Maze* environment.

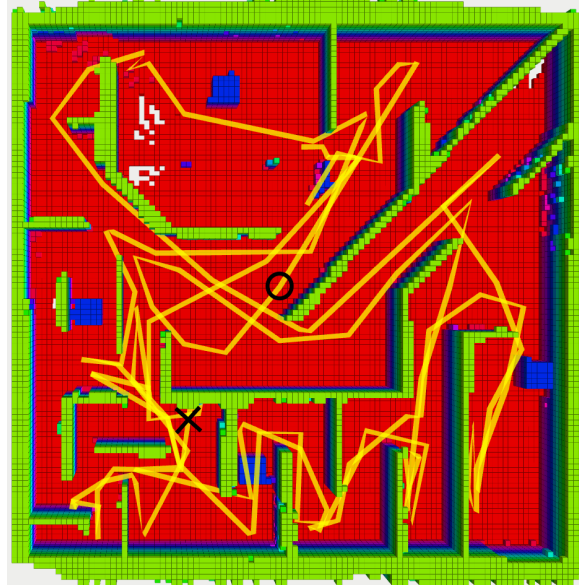


Figure 4.6: Top-down view of a map of the *Maze* environment created by our algorithm. Voxels are coloured based on their height, with red indicating floor voxels and green voxels near the cropped roof. The MAV’s starting position is marked by a black circle, its path by a yellow line and its final position by a black cross.

4.5.3 Powerplant Environment Simulation

In order to evaluate our method in an outdoors scenario, the $33 \times 31 \times 26$ m *Powerplant* environment from the Gazebo simulator [Open Robotics, 2018a] was used. We used a voxel resolution of $r = 0.2$ m and a higher maximum depth range of $d_{\max} = 7$ m. Figure 4.7 [Right] shows the explored volume over time, averaged over 10 runs, for various MAV maximum linear velocities. Figure 4.7 [Left] shows a map created by our algorithm and the corresponding MAV path. In this simulation, the planner is unable to explore the whole space because a large portion of it is unobservable, e.g. the interior of the building. For lower MAV velocities the exploration is slower not only due to the lower movement speed but because candidate views further away have a lower information gain over time due to the increased time it takes to fly to them. Thus the MAV more thoroughly explores the nearby regions before moving to ones further away. The same environment has been used to evaluate [Cieslewski et al., 2017] and [Selin et al., 2019] by reporting only the time required to complete the exploration. However, since both [Cieslewski et al., 2017] and [Selin et al., 2019] use different exploration termination conditions, a comparison of the completion time would not necessarily be indicative of a plan-

ner's performance. A more meaningful comparison could be made by comparing the amount of volume explored in a given amount of time, as presented in Figure 4.7 [Right].

It can be observed that the MAV paths shown in Figures 4.6 and 4.7 are longer than necessary and involve some revisits of previously mapped areas, mainly due to the greedy nature of the proposed exploration planner. Its performance can be improved by extending the planner to consider the MAV's past path or by using a hierarchical planning scheme. In hierarchical planning, a high-level planner would select the subset of the environment to explore next and the proposed planner would be used to explore the subset, switching back to the high-level planner as necessary.

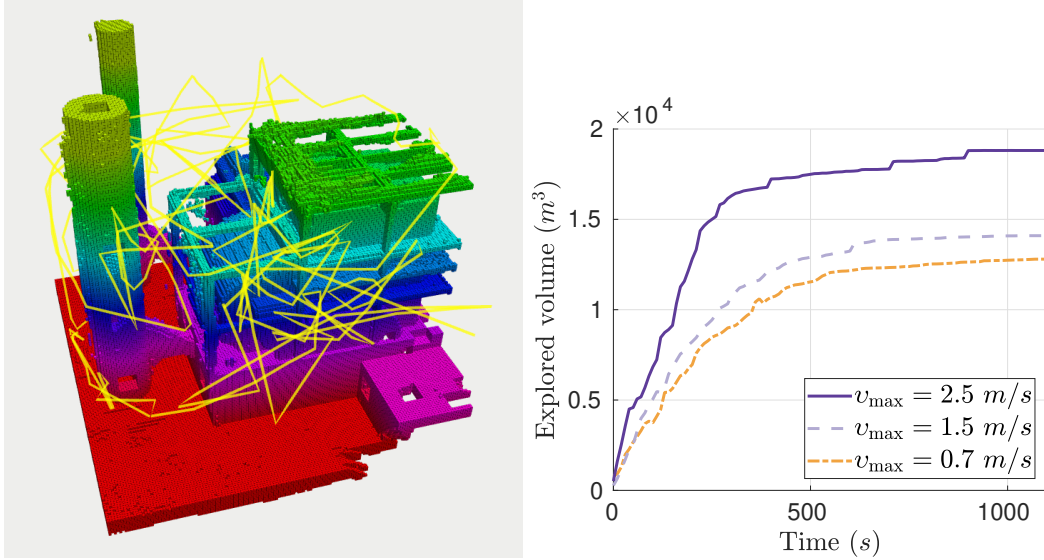


Figure 4.7: [Left] A map of the *Powerplant* environment created by our method. Voxels are coloured based on their height and the MAV's path through the environment is marked by a yellow line. [Right] *Powerplant* environment explored volume over time.

4.5.4 Computational Complexity

Since the planner is running on-board an MAV its computational complexity is an important characteristic. Sampling the candidate positions has a complexity of $\mathcal{O}(N_c)$ and computing a single collision free path with N_p points is $\mathcal{O}\left(\frac{N_p-1}{r^3}\right)$. Performing a single sparse raycast with n horizontal rays and m vertical rays scales

with $\mathcal{O}\left(\frac{nm}{r}\right)$ while the yaw optimisation scales with $\mathcal{O}(n)$. Computing the utility of a single path requires computing its length, thus it scales with $\mathcal{O}(N_p)$. Finally, for the best path only, the yaw at the intermediate path points is computed with a sparse raycast and yaw optimisation for each point giving $\mathcal{O}\left(N_p\left(\frac{nm}{r} + n\right)\right)$. Putting it all together, the computational complexity for a single planner iteration is $\mathcal{O}\left(N_c\left(\frac{N_p}{r^3} + \frac{nm}{r} + n + N_p\right) + N_p\left(\frac{nm}{r} + n\right)\right)$.

For a quantitative comparison to other methods the per-iteration planner computation time for all simulations was measured. The mean and standard deviation for the three simulated environments at different map resolutions for both our planner and *RH-NBVP* are presented in Table 4.2. The reduction in computational time at higher resolutions can be attributed to the sparse raycasting, the efficient collision checking performed by our method. It should be noted that receding horizon planners like [Bircher et al., 2018] and [Selin et al., 2019] typically perform a large number of planning iterations since they perform only a portion of the computed path at each iteration. Our algorithm performs relatively few planning iterations since each computed path is followed until its end before replanning.

Voxel resolution r (m)	<i>Apartment</i> (ms)	<i>Maze</i> (ms)	<i>Powerplant</i> (ms)
Ours	0.4	122 ± 36	—
	0.2	156 ± 109	152 ± 20
	0.1	68 ± 27	238 ± 80
<i>RH-NBVP</i>	0.4	73 ± 8	—
	0.2	707 ± 44	775 ± 50
	0.1	7940 ± 410	8540 ± 425

Table 4.2: Per-iteration computation time.

4.5.5 Real World Experiments

A real world experiment was performed in order to demonstrate the feasibility of running the proposed exploration algorithm on-board an MAV. The experiment was conducted in a 7×5.5×5 m room equipped with a Vicon motion capture system providing the MAV pose. The MAV used was a DJI F550 hexacopter equipped with an ASUS Xtion Pro RGB-D camera. The entire system, both mapping and exploration planning was run on-board the MAV on an Intel NUC with an Intel Core i7-7567U CPU operating at 3.5 GHz. After a manual take-off to a height of 1 m by the safety pilot, the exploration algorithm is initiated. A 0.5 × 1.3 × 0.8

m obstacle was placed inside the room in order to create additional frontiers and force the MAV to navigate around it. The mean per-iteration computation time was 383 ms with a standard deviation of 144 ms. This time is higher compared to the simulated experiments due to the different CPU used. The resulting map can be seen in Figure 4.8 [Left] and a photograph of the experimental setup in Figure 4.8 [Right].

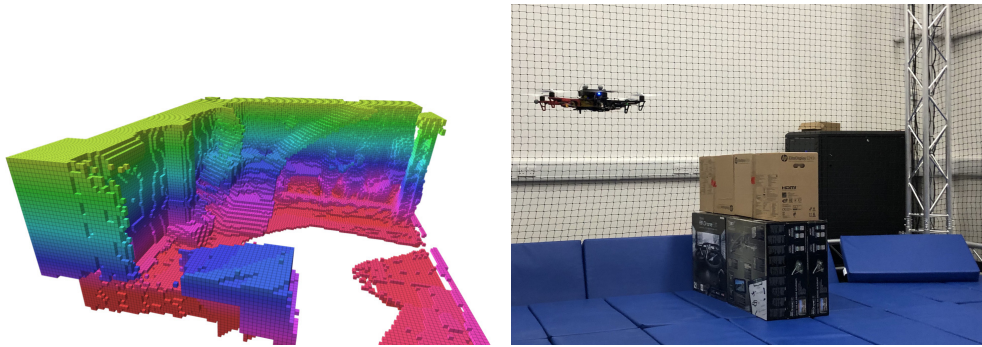


Figure 4.8: [Left] Map of the room the experiment was conducted in. Voxels are coloured based on their height and two of the walls have been cropped for visualisation. [Right] A photograph of the MAV and the obstacle.

4.6 Conclusion

In this chapter, we have presented a fast and computationally efficient exploration strategy that is a hybrid between frontier-based and sampling-based strategies. It was compared in a wide array of simulated environments against a classical sampling-based method, demonstrating faster exploration and reduced computational demands due to higher sample efficiency. A real world experiment was also performed, highlighting the feasibility of deploying the proposed exploration strategy in real-time on-board an MAV.

In Chapter 5, this exploration method is extended to allow semantic, object-centric exploration, while in Chapter 6 it is adapted to use local submaps instead of a monolithic map.

Semantic Object-Centric Exploration

Contents of Chapter

5.1	Introduction	83
5.2	Related Work	84
5.3	Problem Formulation	86
5.3.1	Environment Model	86
5.3.2	MAV Model	86
5.4	Methodology	87
5.4.1	Background Mapping	87
5.4.2	Object Mapping	89
5.4.3	History of Incomplete Depth Maps	90
5.4.4	Exploration Planning	91
5.5	Evaluation	94
5.5.1	MAV Simulator	94
5.5.2	Simulated Experiments	94
5.5.3	Real World Experiments	98
5.6	Conclusion	99

Parts of this chapter appear in:

Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas, Christopher Choi, Binbin Xu and Stefan Leutenegger (2023). **Finding Things in the Unknown: Semantic Object-Centric Exploration with an MAV**. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Papatheodorou et al., 2023]

5.1 Introduction

Exploring an unknown environment using an autonomous robot has been well-researched. The objective is typically formulated as uncovering as much free space as possible in a given time-frame, or uncovering all the limited free space as fast as possible [Yamauchi, 1997, Bircher et al., 2018, Dai et al., 2020, Duberg and Jensfelt, 2022]. To achieve this, a tight interaction between free space mapping, planning, and control is crucial. MAVs are common platforms for autonomous exploration due to their ability to quickly move in any direction, allowing them to explore complex 3D environments.

However, exploration is not often a goal in and of itself, it is typically a necessary step in solving a larger task. We believe that many robotics tasks that require exploration also require object-level and semantic understanding of the environment. Although there has been work aimed at exploring an unknown environments while detecting objects of interest, their goal is often finding a single object instance [Chaplot et al., 2020, Zeng et al., 2020] or is not focused on object reconstruction [Dang et al., 2018a].

Significant progress in the fields of autonomous exploration and object detection was made during the DARPA Subterranean Challenge (SubT) [DARPA, 2021]. The goal of the SubT challenge was to deploy robots to autonomously map and navigate underground environments in order to detect specific objects of interest. Due to the different nature of the objects of interest, a combination of visual methods — such as YOLOv3 [Redmon and Farhadi, 2018] — gas sensors, and WiFi/Bluetooth signal strength measurements was employed [Tranzatto et al., 2022]. However, most methods employed in SubT do not take object detection into account during exploration planning [Williams et al., 2020, Hudson et al., 2022, Rouček et al., 2019], relying instead on incidental object observations during the course of purely geometric exploration. While a semantic map is maintained in [Agha et al., 2021], it is the robot operator’s responsibility to direct exploration

in regions that have been identified as containing objects of interest. In [Best et al., 2022] the candidate next viewpoints generated by the exploration planner are poses where it is predicted that the robot will observe currently unmapped surfaces. In addition, since the SubT challenge only required localising objects, none of the methods employed focus on high-quality object reconstruction.

In this paper we propose a task-specific exploration method aimed at finding all objects of interest in an unknown environment and reconstructing them with high accuracy. To the best of our knowledge, this is the first work using object-level maps for active object reconstruction and exploration. In summary, we propose the following contributions:

- A mapping pipeline suitable for exploration, path planning and high quality object reconstructions.
- An exploration utility function tailored to discovering objects and creating high quality object reconstructions.
- A scheme that accounts for incomplete depth maps preventing the exploration algorithm from getting stuck in regions of the scene where depth measurements consistently cannot be obtained.
- An open source MAV exploration simulator based on Habitat [Savva et al., 2019] and the Robot Operating System (ROS) ⁴.

5.2 Related Work

The SLAM community has moved its attention from pure geometric representations to semantically annotated reconstructions as in *SemanticFusion* [McCormac et al., 2017]. One approach is to build object-centric maps for each detected object in the scene. *SLAM++* [Salas-Moreno et al., 2013] is an early object-level mapping system that matches observed objects to pre-scanned shapes. Later, works such as *Fusion++* [McCormac et al., 2018] and *Kimera* [Rosinol et al., 2020a] were proposed to incrementally build object-level maps inside a dense SLAM system. Object-level representations in a SLAM system have the potential to provide more reliable loop

⁴<https://github.com/smartroboticslab/semantic-exploration-icra-2023>



Figure 5.1: Objects reconstructed using our method [Top left] and their corresponding ground-truth meshes provided by the Matterport3D dataset [Chang et al., 2017] [Bottom left]. Top-down view of 3D reconstruction after exploration, and MAV path in yellow [Right].

closure detection [McCormac et al., 2018] and robustness to dynamic objects [Xu et al., 2019].

Powered by these advancements, several contemporary works propose the inclusion of semantic information for autonomous exploration in unknown environments. In [Dang et al., 2018a, Ashour et al., 2020], the authors propose a method to create a semantically-annotated mapping system and include the re-observation of detected objects in the utility function for the proposed sampling-based path planning algorithm. [Zeng et al., 2020] further take inter-object spatial relationships into consideration in the next-best-view selection by maintaining the belief over the location of key objects. In terms of information-based exploration, [Ashgarivaskasi and Atanasov, 2021] develops a Bayesian multi-class semantic mapping system, where a closed-form lower bound for Shannon mutual information is computed to evaluate an optimal trajectory. Instead of using local goal detection and heuristic utility functions, learning-based approaches have also been proposed to train a navigation policy network in the semantic map via reinforcement learning [Chaplot et al., 2020]. The goal of these semantically-guided exploration methods is either to find one object of a specific class or finding all objects in the environment as a result of classic exploration. They do not focus on reconstruct-

ing the found objects in detail nor do they employ an exploration algorithm to facilitate the detection of all objects in the environment. It is these gaps that our proposed method aims to fill.

5.3 Problem Formulation

The goal of this work is to use a sensor-equipped MAV to explore and map an unknown space and, at the same time, find and reconstruct in detail an unknown number of objects. Thus, the MAV must be able to map its environment, plan safe exploration paths, detect objects and create separate object reconstructions.

5.3.1 Environment Model

We model the static environment as a bounded volume $V \subset \mathbb{R}^3$. Due to the geometry of the environment and the MAV, as well as the sensor's mounting pose, there can be points $V_{\text{unob}} \subset V$ that cannot be observed by the sensor, limiting exploration to the observable subset $V_{\text{obs}} = V \setminus V_{\text{unob}}$. Thus autonomous exploration creates a map $M : V_{\text{obs}} \rightarrow [0, 1]$, a field of occupancy probabilities over V_{obs} , with the goal of minimising the field entropy. This means updating M to reflect which regions of V_{obs} are *free* and which ones are *occupied*. If no prior information about the environment is available, everything is initially *unknown*, $M(\mathbf{v}) = 0.5, \forall \mathbf{v} \in V_{\text{obs}}$. Frontiers are defined as the boundaries between *free* and *unknown* space [Yamauchi, 1997] and identify regions that will extend the map when observed. The map M can be used to plan collision-free paths for the MAV to follow.

It is assumed that the environment contains $N_o \in \mathbb{N}$ static objects of interest, each one assigned a semantic class from the set of semantic classes $\mathcal{C}$. Each semantic class has an associated target map resolution $r_c, c \in \mathcal{C}$ selected by the user. The goal of object-centric exploration is to find all objects and create a high-quality map $M_i^c, i \in \{1 \dots N_o\}, c \in \mathcal{C}$ with a resolution at least as fine as r_c for each one.

5.3.2 MAV Model

For the purposes of exploration and path planning, the MAV's state $\mathbf{x}$ consists of its position vector $\mathbf{r} = [x, y, z]^T \in V$ in the world frame $\underline{\mathcal{F}}_W$ and a yaw angle scalar, $\psi \in [-\pi, \pi)$ with respect to the origin of the world coordinate, thus $\mathbf{x} =$

$[x, y, z, \psi]^T \in V \times [-\pi, \pi)$. This is the 4-DoF subset of the MAV's configuration space that is controllable while hovering, as described in Section 2.3. We assume that the MAV has a maximum linear velocity $v_{\max} \in \mathbb{R}^+$ and a maximum yaw rate $\omega_{\max} \in \mathbb{R}^+$ and is enclosed in a sphere of radius R centred at $\mathbf{r}$. While our planning does not consider the MAV's dynamics or roll and pitch angles, the full 6 degree of freedom pose is needed for mapping. It is expressed as a rigid body transformation matrix T_{WC} from the MAV camera frame $\underline{\mathcal{F}}_C$ to the world frame $\underline{\mathcal{F}}_W$ and can be estimated using a SLAM system.

The MAV is equipped with an RGB-D sensor of resolution $W \times H$, focal lengths f_x and f_y , and depth range $[d_{\min}, d_{\max}] \subset \mathbb{R}^+$ inclusive. The sensor produces synchronised pairs of colour and depth images $\mathbf{C}$ and $\mathbf{D}$ respectively.

Each colour image $\mathbf{C}$ has a number of corresponding object instance segmentation masks $\mathbf{S}_k$, $k \in \{1 \dots N_d\}$ where $N_d \in \mathbb{N}$ is the number of detected objects.

5.4 Methodology

Our approach consists of a mapping module and a planning module. The mapping module receives the current camera pose T_{WC} , depth and colour images and object instance masks and produces a background map and associated frontiers as well as individual object maps. The planning module receives the current pose, the background map and its frontiers and the individual object maps and produces the path to the MAV's next goal pose. The mapping module is run as often as possible using the latest measurements while the planning module is run each time the MAV reaches its goal pose. Thus, the background and object maps are being continuously updated as the MAV moves to its next goal pose. The required initial information consists of the semantic classes of interest, their associated resolutions and enough *free* space around the MAV's initial position to allow planning paths. Figure 5.2 shows a diagram of the pipeline.

5.4.1 Background Mapping

The background map with voxel resolution $r \in \mathbb{R}^+$ is created using a modified version of the multi-resolution occupancy mapping pipeline from [Funk et al., 2021] with some extra information stored in the map compared to the method described in Chapter 4. In [Funk et al., 2021] an octree-based volumetric occupancy map is

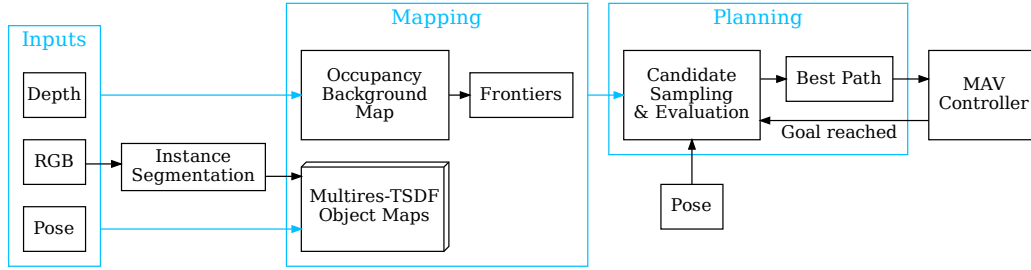


Figure 5.2: Diagram of the proposed approach. The mapping module receives depth and colour image pairs and the corresponding instance segmentation and pose to update the background and individual object maps and, the set of frontiers. The planning module receives the background and object maps, the set of frontiers and, the current pose to produce the next goal path. Planning happens again when the MAV has completed the goal path.

created in real-time from posed depth images. Path planning can be performed directly on the map due to its explicit free space representation and the fact that occupancy data is propagated to higher levels of the octree, allowing for efficient occupancy queries.

Frontier Detection

As in Chapter 4, the mapping pipeline is modified to maintain a set of frontiers $\mathcal{F}$. After each map update, a set of candidate frontiers is generated by forming the union of any previously known frontiers with the set of updated octree nodes. These candidate frontiers are then checked for frontiers at the finest allocated level to produce the new set of frontiers $\mathcal{F}$.

Observed Distance Fusion

An addition compared to Chapter 4 is storing the minimum distance each octree leaf node has been observed from. This allows finding map regions that have not been closely observed as described in Section 5.4.4.

Colour Fusion

The final addition to the map compared to Chapter 4 is integrating colour information in each voxel, resulting in more realistic-looking reconstructions. The colour

update is performed as a weighted average using the same weight as for the occupancy probability.

Map Update

The background map M is updated as quickly as possible using pairs of depth and colour images and their corresponding poses without filtering out any detected objects. This allows performing path planning without having to consider the individual object maps since the background map already contains this information.

5.4.2 Object Mapping

It is assumed that an object segmentation method, providing per-pixel object instance masks given a colour image, is available. Notable such methods include *Mask R-CNN* [He et al., 2017] and *YOLACT* [Bolya et al., 2019].

Object mapping was performed using a method similar to *MID-Fusion* [Xu et al., 2019], which has demonstrated the feasibility of using *Mask R-CNN* to perform object-level mapping. The modifications made to *MID-Fusion* were aimed at improving performance, memory usage and reconstruction quality. We changed the underlying map representation to a modified version of the adaptive-resolution octree-based TSDF mapping from [Vespa et al., 2019] which offers improved performance and reduces aliasing artefacts. A TSDF representation was chosen for objects because of its reduced computational requirements compared to occupancy representations.

Object Matching

Similar to [Xu et al., 2019], objects detected in the colour image $\mathbf{C}$ are matched to known objects in the map, after instance masks smaller than a threshold are discarded to prevent unrealistic detections in synthetic datasets. Each known object is first raycasted from the current camera pose T_{WC} to create a per-object instance mask, taking occlusions from the background and other objects into account. These raycasted instance masks represent the parts of known objects visible from the current pose. For each detected object instance mask $\mathbf{S}_k$, its intersection over union (IoU) with each per-object raycasted instance mask is computed. The detected object is associated with the known object for which the highest mask

IoU above a certain threshold was achieved. If there is no IoU above the threshold then a new map is created for the detected object. While in [Xu et al., 2019] the object map resolution was constant, in our method it depends on the object’s semantic class, allowing efficient mapping of objects of a wide range of sizes.

Object Map Update

As in [Xu et al., 2019], we integrate colour and foreground probability information in each object map voxel using a weighted average update. In the case of visible known objects that were undetected in the current colour image, depth and colour information is integrated into them using their raycasted instance masks while omitting the update of foreground probabilities. Compared to [Xu et al., 2019], we also keep track of the minimum observed distance as in the background map.

Memory Usage Reduction

We reduce the memory footprint of each voxel by using 16-bit and 8-bit fixed-point numbers for the TSDF value and foreground probability, respectively. This reduces the required memory per voxel by 39% with negligible degradation of reconstruction accuracy.

5.4.3 History of Incomplete Depth Maps

Depth images often contain invalid or missing data due to e.g. reflective surfaces, occlusions or missing geometry in the case of synthetic datasets. A depth image with a large amount of missing data can cause a naive exploration algorithm to get stuck expecting a reduction in map entropy that never happens due the amount of missing data. To tackle this issue, we store the history of invalid depth measurements as a low-resolution 3D grid over V containing binary 360° images. All images are initialised to 1 denoting valid depth. For each depth image $\mathbf{D}$ integrated into the background map, the history image $\mathbf{H}$ corresponding to pose T_{WC} is fetched. The 2D coordinates of each invalid measurement in $\mathbf{D}$ are back-projected in 3D and then re-projected onto $\mathbf{H}$, setting the corresponding pixel to 0. Both the 3D grid and image resolution can be quite coarse, avoiding any significant performance degradation. A grid resolution of 0.5 m and an image resolution of 36×10 were found to be sufficient in practice.

5.4.4 Exploration Planning

Our algorithm is a hybrid between sampling-based and frontier-based exploration. Candidate next poses are sampled close to the frontiers and the known objects, ranked based on the proposed utility function, and the highest ranked pose is selected as the next goal. Compared to the method described in Chapter 4, in this approach

- candidate next positions are also sampled close to objects instead of only close to frontiers,
- the exploration utility is computed using two more raycasts in addition to the entropy raycast described in Chapter 4, and
- the invalid depth history described in Section 5.4.3 is used to prevent the exploration from becoming stuck.

Candidate Next Position Sampling

Candidate next positions $\hat{\mathbf{r}}_j \in V$, $j \in \{1 \dots n\}$, $n \in \mathbb{N}^+$ are sampled among the set of frontiers and the known objects without replacement until n candidates have been sampled or no more frontiers or objects are left to sample, as shown in Figure 5.3 [Left]. Sampling at frontiers favours exploring unknown space while sampling at objects allows obtaining higher quality object observations. Sampling can be biased towards frontiers or objects using the frontier sampling probability P_{fr} to strike a balance between object reconstruction quality and exploration.

Path Planning to Candidate Next Positions

For each sampled candidate position, a path $P_j(\mathbf{r}, \hat{\mathbf{r}}_j)$ to it is planned from the MAV's current position, as shown in Figure 5.3 [Left]. The path is planned on the background map using the Informed RRT* [Gammell et al., 2014] implementation from the Open Motion Planning Library [Sucan et al., 2012]. Since sampled positions are at or near non-*free* space, paths are planned as close to them as possible with no requirement for a complete solution.

Raycasting at Candidate Next Positions

In order to evaluate each candidate next position, three low-resolution, $w \times h$, 360° raycasts are performed from each candidate position $\hat{\mathbf{r}}_j$. The raycasting resolution $w \times h$ is independent of the input image resolution $W \times H$ and typically much smaller to reduce the computational requirements [Oleynikova et al., 2018a]. This results in three gain images with values in the interval $[0, 1]$ inclusive. The 360° raycasting rays from a single candidate are shown in Figure 5.3 [Middle].

An entropy raycast of the background map M is performed, resulting in an entropy gain image $\mathbf{G}_{ent}$. Each pixel of $\mathbf{G}_{ent}$ contains the normalised sum of the Shannon entropy of each voxel $\mathbf{v}$ along the corresponding ray. Entropy is accumulated along each ray until $d_{\max}$ or an *occupied* voxel is reached and the sum is normalised with its maximum possible value $\frac{(d_{\max} - d_{\min})}{r}$. The entropy gain image guides the exploration towards observing *unknown* space resulting in expansion of the map.

A background gain image $\mathbf{G}_{bg}$ is created by raycasting the minimum observed distance information stored in the background map. Each pixel of $\mathbf{G}_{bg}$ contains the distance gain of the octree leaf node $\mathbf{n}$ containing the first *occupied* voxel hit by the corresponding ray. The distance gain of $\mathbf{n}$ is

$$G(\mathbf{n}) = \begin{cases} 0, & \text{if } d_{\text{node}}(\mathbf{n}) \leq d_{bg} \vee d_{\text{node}}(\mathbf{n}) \leq d_{\text{exp}}(\mathbf{n}) \\ \frac{d_{\text{node}}(\mathbf{n}) - \max(d_{\text{exp}}(\mathbf{n}), d_{bg})}{d_{\max}}, & \text{otherwise} \end{cases}, \quad (5.1)$$

where $d_{\text{node}}(\mathbf{n}) \in \mathbb{R}^+$ is the minimum distance $\mathbf{n}$ has been observed from, $d_{\text{exp}}(\mathbf{n}) \in \mathbb{R}^+$ is the expected observed distance of $\mathbf{n}$ from $\hat{\mathbf{r}}_j$ and $d_{bg} \in \mathbb{R}^+$ is the desired observed distance for the background map. The desired observed distance d_{bg} is set to a value that ensures observed objects will appear large enough in a colour image taken from d_{bg} or closer that they can be detected. The background gain image helps ensure all of the environment has been observed from close enough that all objects have been detected.

Similarly to the background gain image, an object gain image $\mathbf{G}_{obj}$ is computed by raycasting the minimum observed distance stored in all object maps while taking occlusions into account. The per-pixel gain is computed in the same manner as for the background except that the object desired observed distance $d_{obj} \in \mathbb{R}^+$ is used instead of d_{bg} in Equation (5.1). The desired observed distance $d_{obj} < d_{bg}$ for objects of all semantic classes is set to a value that minimises the distance-based

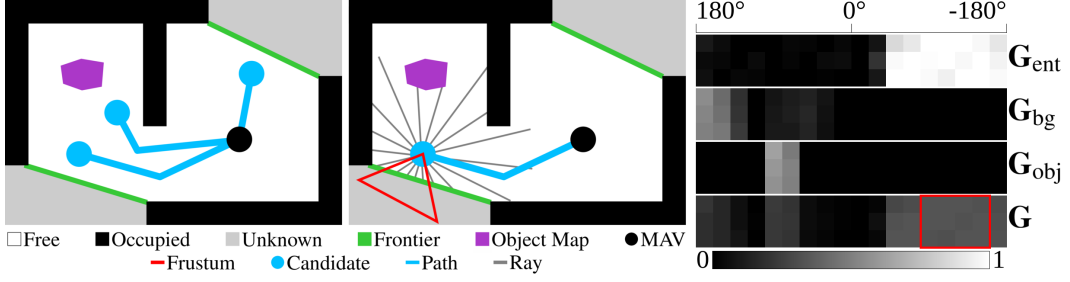


Figure 5.3: [Left] Candidate view sampling near frontiers and objects, and path planning to candidates from the current pose. [Middle] Sparse 360° raycasting from one of the candidate views and optimal yaw frustum in red. [Right] Entropy, background, object and combined gain images from the raycast shown in [Middle] with the optimal yaw field-of-view in red.

depth sensor noise while taking $d_{\min}$ and the MAV dimensions into account. The object gain image results in the MAV getting closer, higher-quality observations of existing objects.

Candidate Next Pose Evaluation

A single gain image $\mathbf{G}$ is produced from a weighted sum of the individual gain images and masking by the invalid depth history

$$\mathbf{G} = (\alpha_{ent}\mathbf{G}_{ent} + \alpha_{bg}\mathbf{G}_{bg} + \alpha_{obj}\mathbf{G}_{obj}) \circ \hat{\mathbf{H}}, \quad (5.2)$$

where $\alpha_{ent}, \alpha_{bg}, \alpha_{obj} \in [0, 1]$, $\alpha_{ent} + \alpha_{bg} + \alpha_{obj} = 1$ are weighting factors, $\hat{\mathbf{H}}$ is the invalid depth history image corresponding to position $\hat{\mathbf{r}}_j$ and $\circ$ denotes the element-wise matrix product. The weighting factors are used to solve the object-level exploration-exploitation dilemma by balancing between exploring, detecting new objects and improving the quality of existing objects. A sliding window on the gain image $\mathbf{G}$ is used to compute the yaw angle ψ_j at which the maximum gain g_j is achieved, similarly to [Selin et al., 2019]. Some example gain images and the corresponding optimal yaw angle are shown in Figure 5.3 [Right]. The utility u_j of each candidate j is computed to maximise the gain over time as $u_j = g_j / t_j$, where the time t_j required for the MAV to complete the path to candidate j is estimated based on the assumption that it always flies at speed $v_{\max}$ and rotates at speed $\omega_{\max}$.

Next Goal View

The candidate $G \in \{1 \dots n\}$ with the highest utility is selected as the next goal. A yaw angle needs to be assigned to each vertex of the goal path P_G . The first vertex, which coincides with the current MAV position, gets assigned the current yaw while the final vertex gets assigned the optimal yaw ψ_G . The yaw of intermediate path vertices is computed similarly to the final vertex yaw, by performing 360° raycasts and computing the optimal yaw angle.

Termination Condition

The exploration stops when $\mathcal{F} = \emptyset$, and all of the background and objects have been observed from a distance at least d_{bg} and d_{obj} , respectively.

5.5 Evaluation

5.5.1 MAV Simulator

To quantitatively evaluate the proposed method and make our results reproducible, we created an MAV simulator suitable for object-centric exploration. The MAV dynamics are simulated using the prediction step of the MAV Model Predictive Controller from [Tzoumanikas et al., 2020b]. The simulator uses Habitat [Savva et al., 2019] to provide colour and depth images as well as ground-truth object segmentation masks rendered from the MAV poses. We included additive Gaussian, zero-mean noise with a standard deviation based on distance in the synthetic depth images to make them more closely resemble those produced by a RealSense D455 sensor. The noise standard deviation for a depth measurement d was computed as $\sigma(d) = \max(0.005, \min(0.002 d, 0.2))$.

5.5.2 Simulated Experiments

All simulated experiments were run on a computer with an Intel Core i7-6700K CPU, 16 GB of memory and an NVIDIA RTX 3080 GPU. They were run on Ubuntu 20.04 using ROS Noetic [Open Robotics, 2020] and compiled with GCC 9.4.0 using the O3 optimisation level. The parameters used for the simulated experiments can be found in Table 5.1.

Parameter		Value
Map/object resolution	r, r_{chair}	0.04 m, 0.02 m
MAV radius	R	0.125 m
MAV maximum linear velocity	v_{max}	1.5 m/s
MAV maximum angular velocity	ω_{max}	0.75 rad/s
Sensor resolution	$W \times H$	320×240
Sensor measurement range	$[d_{\text{min}}, d_{\text{max}}]$	$[0.1, 10]$ m
Sensor focal lengths	f_x, f_y	262.5 px
Candidate poses sampled	n	20
Raycasting resolution	$w \times h$	36×10
Frontier sampling probability	P_{fr}	0.5
Background/object desired distance	$d_{\text{bg}}, d_{\text{obj}}$	3 m, 1 m
Entropy/background/object weights	$\alpha_{\text{ent}}, \alpha_{\text{bg}}, \alpha_{\text{obj}}$	0.34, 0.33, 0.33

Table 5.1: Simulated experiment parameters.

The proposed semantic exploration method is compared against a classic exploration method, similar to the one from Chapter 4 with the addition of the object mapping and the invalid depth history described in Sections 5.4.2 and 5.4.3 respectively. The classic exploration method is obtained by setting $\alpha_{\text{ent}} = 1$, $\alpha_{\text{bg}} = \alpha_{\text{obj}} = 0$ and $P_{fr} = 1$. This is an ablation study of the importance of sampling candidates near objects and the two distance gain images for efficient object-centric exploration. It also helps showcase that the proposed method retains competitive exploration performance.

Both methods are evaluated on the Matterport3D [Chang et al., 2017] dataset which consists of semantically annotated 3D scans of real world interior spaces. Since this dataset is collected from real-world data, there is noise and holes in the ground-truth depth and segmentation masks. The evaluation consists of 5 runs of each method, classic and semantic, on sequences 1LXtFkjw3qL, 29hnd4uzFmX, 2azQ1b91cZZ, 2n8kARJN3HM, 2t7WUuJeko7, and 8WUmhLawc2A. These sequences were chosen because they have relatively little missing data while offering interesting spaces to explore. Chairs were chosen as the semantic object class of interest due to their abundance in the used sequences. Some reconstructed objects and their corresponding ground-truth meshes are shown in Figure 5.4.

Figure 5.5 [Left] shows the explored volume over time. It can be observed that the performance of the proposed method is on-par with classic exploration.

Figure 5.5 [Right] shows the percentage of objects detected over time. The

5. Semantic Object-Centric Exploration



Figure 5.4: Objects reconstructed using the proposed method [Top] and their corresponding ground-truth meshes [Bottom]. Notice some artefacts due to erroneous segmentation masks.

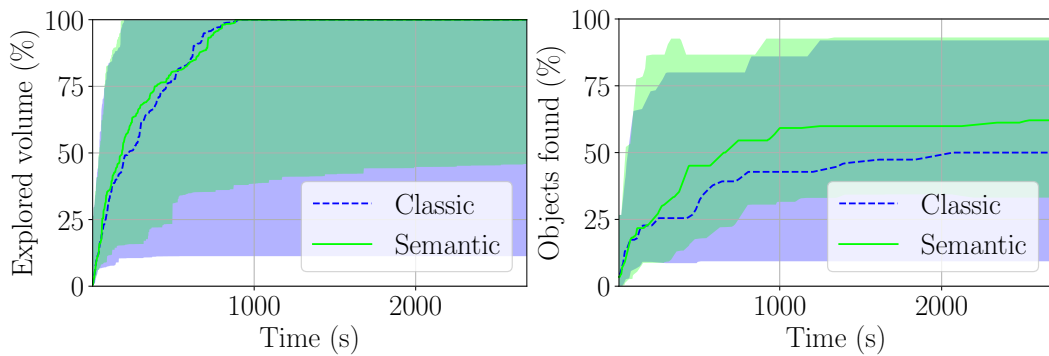


Figure 5.5: Median, 10th and 90th percentiles of explored volume [Left] and percentage of objects found [Right].

proposed method finds more objects overall, faster and more consistently than the classic approach, indicating the importance of the background distance gain and sampling candidates near objects. Sampling candidates near known objects can help detect new objects quickly because objects are often close to each other, e.g. chairs tend to be close to other chairs. One reason for objects not being detected is partial observations, which lead to high uncertainty in the object detection network. If an object does not cover a large enough part of the image, it cannot be detected. Thus, observing small parts of an object from different perspectives in different images can result in full exploration while the object remains undetected. Another problem arises from the imperfect instance segmentation masks in the Matterport3D dataset, where the mask of one object might bleed onto another object. This can cause the algorithm to erroneously merge the two distinct objects into one, only one of which will be considered matched to a ground-truth object.

We use the (pseudo) ground-truth meshes from Matterport3D to evaluate the meshes generated by our method in terms of accuracy and completeness. Accu-

racy is computed as the root-mean-square error while completeness is computed as the percentage of the ground-truth mesh vertices for which there is a reconstructed mesh vertex within 5 cm. Figures 5.6 and 5.7 show the background and object reconstruction accuracy over time and completeness over time, respectively, aggregated over all runs. It can be seen that both methods achieve a similar level of reconstruction accuracy. There is little variation over time in the background and object accuracy because they are observed from a close enough distance even during classic exploration as Matterport3D consists of interior spaces. The apparent discrepancy between the explored volume and background completeness is due to the fact that the former is a volume metric while the latter is a surface metric. This means that the resulting background mesh often has small holes due to occlusions or insufficient data causing a lower completeness score.

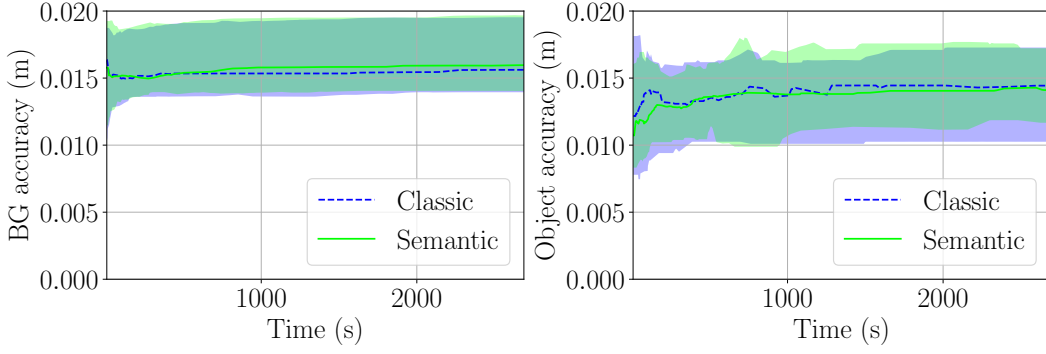


Figure 5.6: Median, 10th and 90th percentiles of background [Left] and object [Right] reconstruction accuracy.

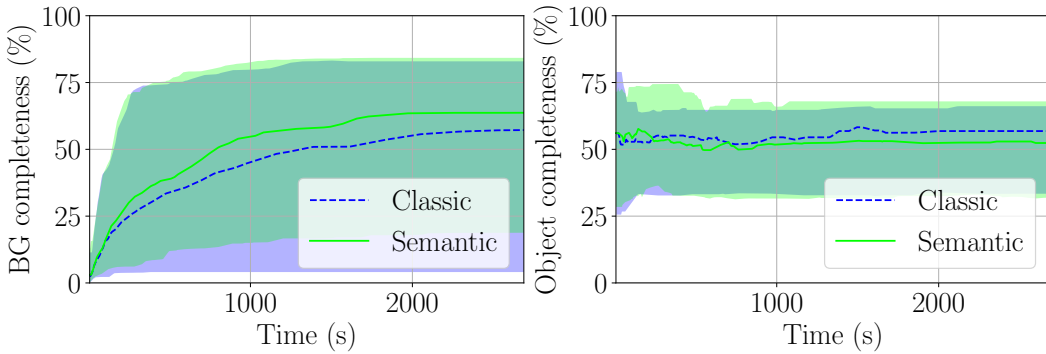


Figure 5.7: Median, 10th and 90th percentiles of background [Left] and object [Right] completeness.

We also evaluate the percentage of the background and objects that were observed from at most the desired distance, d_{bg} and d_{obj} , respectively, as shown in

Figure 5.8. We observe that the proposed method observes a larger part of the environment and objects from the desired distance and in the case of objects, it does so more consistently. The inaccurate semantic segmentation masks in the Matterport3D dataset can result in artefacts in the object reconstructions, e.g. including part of the background. These artefacts can be occluded, preventing them from being observed and reducing the percentage of objects observed from at most d_{obj} .

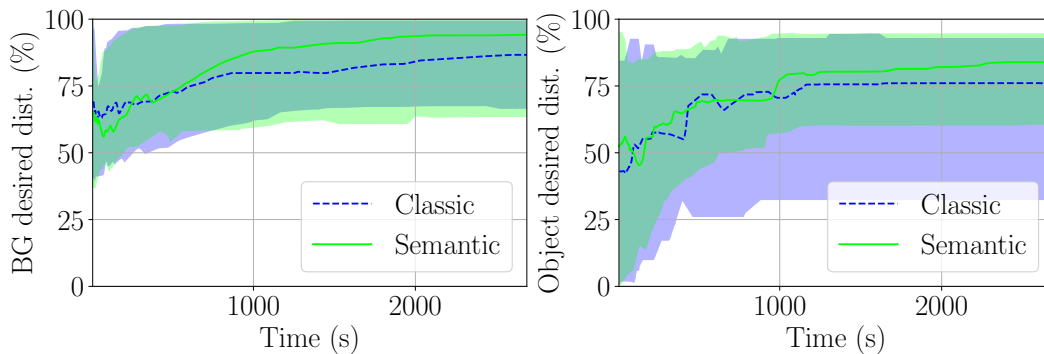


Figure 5.8: Median, 10th and 90th percentiles of the percentage of the background [Left] and objects [Right] observed from at least d_{bg} and d_{obj} respectively.

5.5.3 Real World Experiments

A real world experiment was further conducted in order to showcase the feasibility of using the proposed approach on-board an MAV. The MAV used for the experiments was a DJI F550 hexacopter equipped with an Intel RealSense D455 RGB-D camera. The MAV pose was provided by a Vicon motion capture system. All processing was performed on-board the MAV on an NVIDIA Jetson Xavier NX computer. The experiments were conducted in a $7 \times 5.5 \times 5$ m room with a $3.6 \times 3 \times 1.3$ m volume available to the MAV for safety reasons. The semantic classes used for object reconstruction were limited to backpacks and keyboards. The background map resolution was 4 cm and the target object resolution was 2 cm for all semantic classes. The pre-trained *Mask R-CNN* [He et al., 2017] model from [Abdulla, 2017] was ported to TensorRT to obtain object segmentation masks. *YOLACT* [Bolya et al., 2019] was also considered but it resulted in lower-quality masks compared to *Mask R-CNN* in our qualitative tests. Segmentation was run on the latest colour image not currently being integrated, as fast as possible, on a background thread.

The experiment lasted for 405 s during which 81 frames were integrated and 31 planning iterations took place. The average frame integration time was 1.835 s, the average segmentation time was 1.167 s and the average planning time was 2.803 s. Segmentation information was available for 71 frames. This rather high percentage was achieved because the segmentation time was typically smaller than the integration time so that the segmentation information for the next frame was most often available before the integration of the current frame has finished. During exploration 2 out of 3 backpacks and 2 out of 3 keyboards placed in the environment were detected even though all of them were observed by the camera. This is due to the fact that the deployed *Mask R-CNN* was not fine-tuned on this particular environment and did not detect the objects even though they were visible. A photo of the experiment setup, 3D object reconstructions and respective photos are shown in Figure 5.9.



Figure 5.9: [Left] The real world experimental setup and the MAV. Detected and undetected objects are circled in green and red respectively. [Top right] Meshes of the 2 backpacks and 2 keyboards reconstructed during the experiment. [Bottom right] Photos of the corresponding real world objects.

5.6 Conclusion

In this chapter, we have presented a task-specific exploration pipeline for finding and creating high-quality reconstructions of objects in an unknown environment. Through extensive simulation experiments, it was shown that the proposed method consistently detects more objects of interest, without sacrificing exploration speed. We have also demonstrated that it can be run on-board an MAV in real-world scenes.

Navigation and Exploration with SLAM in the Loop

Contents of Chapter

6.1	Introduction	101
6.2	Related Work	102
6.3	Problem Statement	104
6.3.1	Environment Model	104
6.3.2	MAV Model	104
6.4	System Overview	104
6.4.1	State Estimator	105
6.4.2	Occupancy Mapping	106
6.4.3	Exploration Planning	106
6.4.4	MAV Controller	107
6.5	Global Frontier Computation	107
6.5.1	Local Frontier Computation	108
6.5.2	Global Frontier Computation	108
6.6	Evaluation	109
6.6.1	Simulation Results	109
6.6.2	Real World Experiments	114
6.7	Conclusion	116

Parts of this chapter appear in:

Sotiris Papatheodorou, Simon Boche, Sebastián Barbas Laina and Stefan Leutenegger (2025). **Efficient Submap-based Autonomous MAV Exploration using Visual-Inertial SLAM Configurable for LiDARs or Depth Cameras**. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Papatheodorou et al., 2025]

6.1 Introduction

Autonomous exploration of unknown environments has been an active area of research in mobile robotics and remains a challenge. It is essential for potential real-world robot applications, such as industrial or construction site inspection. MAVs are a popular choice of platform due to their versatility, agility and ability to reach areas inaccessible to humans or ground robots. The objective of exploration is commonly formulated as discovering and mapping an environment as fast as possible [Yamauchi, 1997, Bircher et al., 2018, Dai et al., 2020]. Ideally, to achieve full autonomy, the robot incrementally builds an accurate and complete map of its environment online, solely based on sensor inputs, that contains all information required for navigating through and localising in it. State-of-the-art approaches typically formulate this problem as identifying the next-best-pose that maximises some utility and finding a collision-free path to the goal pose.

SLAM methods fusing complementary sensors are widely used for on-board pose estimation. VI-SLAM, combining visual and inertial measurements, offers high short-term accuracy but suffers from the accumulation of drift [Leutenegger, 2022, Campos et al., 2021, Usenko et al., 2019]. Lately, driven by the availability of low-priced LiDAR sensors, LVI-SLAM has been a topic of extensive research [Shan et al., 2021, Zheng et al., 2022, Boche et al., 2024] in the prospect of reducing drift.

In large-scale scenarios, the inevitable drift can lead to significant deterioration of the map accuracy. One potential mitigation is using local submaps, based on the assumption that drift is negligible within a small region. To maintain global consistency, the relative poses between individually rigid submaps can be updated upon loop closures [Wang et al., 2021a] or incorporated into the state estimation problem [Boche et al., 2024, Reijgwart et al., 2019].

Keeping track of frontiers, the boundaries between observed and unobserved space, becomes challenging when using submaps, as a large number of them may

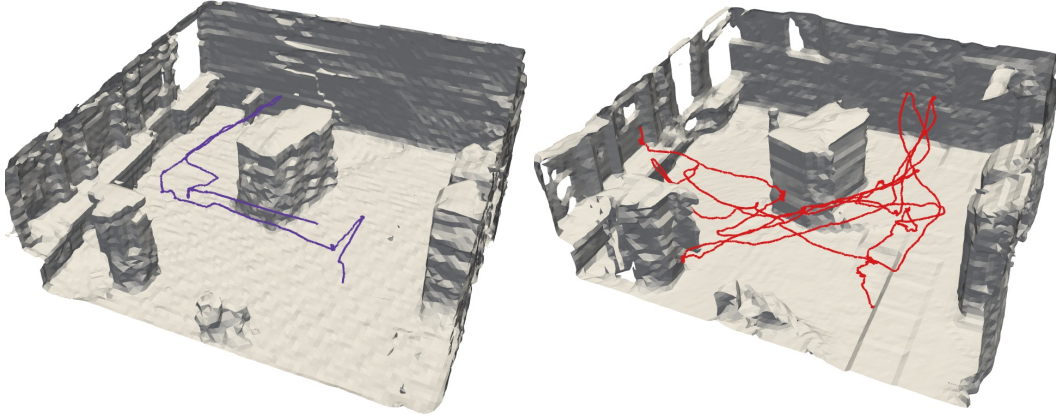


Figure 6.1: MAV trajectory and live-reconstructed mesh at 10 cm resolution for the real-world experiments using a LiDAR [Left] and a depth camera [Right].

be involved. Recent works have proposed hierarchical exploration methods, distinguishing between local and global exploration planning [Schmid et al., 2021, Dang et al., 2019]. In our work, we show that a hierarchical approach is not necessary for fast and efficient exploration.

We propose the following contributions:

- A fast and lightweight 3D exploration framework leveraging submapping and SLAM to account for odometry drift using loop closures.
- An efficient global frontier update method, allowing for accurate and complete large-scale exploration, without having to distinguish between local and global planning.
- The proposed system seamlessly supports MAVs equipped with either depth cameras or LiDAR sensors out-of-the-box.
- Quantitative and qualitative evaluation in simulation and in the real world, showing faster exploration, reduced resource usage, and more consistent and complete reconstructions compared to a state-of-the-art method.

6.2 Related Work

To deal with large-scale environments, recent approaches have proposed hierarchical or multi-stage designs that distinguish between local and global planning.

In [Selin et al., 2019], local planning is handled by an RRT that determines the next best view. The global planner is modelled as a Gaussian process that stores the gain of non-executed viewpoints. *GBPlanner* [Dang et al., 2019] also follows this two-stage pattern and consists of a local RRT*-based planner and a global graph-based planner. The global graph guides the robot towards frontiers, when the local planner reaches a dead-end. Furthermore, [Dang et al., 2019] imposes additional safety constraints on the selected path.

Regardless of the design, accurate live pose estimates are required, typically obtained from SLAM. Several approaches have been proposed to overcome the deterioration of the map quality due to drift, which is especially problematic in large trajectories. [Zhang et al., 2022] addresses this issue by computationally expensive de- and re-integration of measurements upon visual loop closures. Furthermore, exploration is actively steered towards potential loop closure candidates. Another line of works applies submapping strategies. The global map is represented as a collection of submaps which can be re-arranged upon loop closures or pose-graph optimisation, as presented in [Boche et al., 2024, Reijgwart et al., 2019, Millane et al., 2018, Wang et al., 2022].

A successful combination of submapping and hierarchical exploration has been presented in *GLocal* [Schmid et al., 2021]. Submaps based on *Voxgraph* [Reijgwart et al., 2019] are periodically created and updated in a pose-graph optimisation which allows handling loop closures. In *GLocal*, the MAV first explores its surroundings using a local, RRT*-based planner. Once the utility of nearby regions becomes low, the global planner guides the robot towards global frontiers. Traversability graphs from previously completed submaps allow for efficient global path planning.

Just like *GLocal*, our system builds upon the concept of submaps. In contrast to the aforementioned approaches, we do not need to make a distinction between local and global planning. Enabled by efficient updates of the local and global frontiers, a unified global planner can be applied, favouring nearby candidate next poses through the utility function design.

6.3 Problem Statement

In this work, we aim to build an accurate, complete and globally consistent volumetric representation of the environment using a fully autonomous MAV equipped with either a LiDAR or depth camera. To enable scaling to large environments, we represent the environment with submaps.

6.3.1 Environment Model

The static environment is modelled as a bounded volume $V \subset \mathbb{R}^3$. Due to the environment's geometry as well as the MAV and sensor characteristics, there can be points $V_{\text{unob}} \subset V$ that are unobservable, limiting exploration to the observable subset $V_{\text{obs}} = V \setminus V_{\text{unob}}$. Exploration creates a collection of submaps $\mathcal{M} = \{M_i : V_{\text{obs}}^i \rightarrow [0, 1], V_{\text{obs}}^i \subseteq V, i \in \{1 \dots n_m\}\}$ so that $\bigcup_{i=1}^{n_m} V_{\text{obs}}^i = V_{\text{obs}}$, creating a field of occupancy probabilities over V_{obs} with the goal of minimising its entropy. This means updating all M_i to reflect which regions of V_{obs} are *free* and which ones are *occupied*.

6.3.2 MAV Model

The MAV state consists of the position $\mathbf{r} \in V$, orientation $\mathbf{q} \in SO(3)$, and linear velocity $v \in \mathbb{R}^3$. For exploration and path planning we consider a portion of the MAV's full state, $\mathbf{x} = [\mathbf{r}^T \psi]^T \in V \times [-\pi, \pi)$, where ψ is its yaw angle with respect to the world frame $\underline{\mathcal{F}}_w$. This is the 4-DoF subset of the MAV's configuration space that is controllable while hovering, as described in Section 2.3. We also assume the MAV to be enclosed in a sphere of radius $R \in \mathbb{R}^+$ centred at $\mathbf{r}$, and to have a maximum linear velocity $v_{\text{max}} \in \mathbb{R}^+$ and a maximum yaw rate $\omega_{\text{max}} \in \mathbb{R}^+$. The MAV is equipped with a LiDAR sensor or a depth camera with horizontal and vertical field of view $\alpha_h \in (0, 2\pi]$ and $\alpha_v \in (0, \pi]$ respectively, and a measurement range $[d_{\text{min}}, d_{\text{max}}] \subset \mathbb{R}^+$.

6.4 System Overview

Our system combines VI-SLAM, dense 3D occupancy mapping, exploration planning and an MAV MPC. To deal with the accumulated VI-SLAM drift and achieve

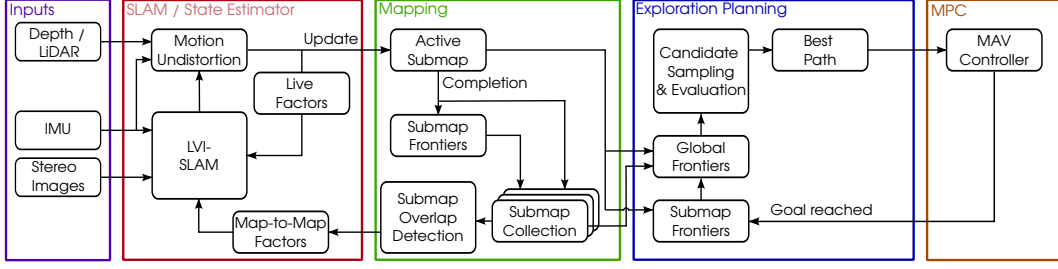


Figure 6.2: A schematic overview of the overall system. We can divide the system into four main modules: (i) a Visual-Inertial SLAM system, optionally with LiDAR, that estimates the MAV’s state based on sensor inputs and submaps, (ii) a dense volumetric mapping backend that divides the global volume into spatially bounded submaps and keeps track of a submap collection, (iii) an exploration planning module which computes the next best path based on global frontiers of all submaps and (iv) a linear MPC that executes trajectories selected by the exploration planner.

locally consistent and accurate maps, we employ spatially bounded submaps. Using SLAM-estimated poses, LiDAR measurements or depth images are integrated into the currently active submap. Upon completion, submaps and their local frontiers are saved in a submap collection. Global frontiers are only updated as needed, using the submap-local frontiers. This allows using a single, global exploration planner, making it unnecessary to distinguish between local and global planning, as other methods do. The global frontiers are used in a sampling-based method to determine the next best path from an information gain perspective, which is then tracked by the MPC. Our method does not require a GPU, allowing its deployment on small and resource-constrained platforms. In the following we describe the individual system modules, shown in Figure 6.2.

6.4.1 State Estimator

We use *OKVIS2* [Leutenegger, 2022] as our state estimator, a state-of-the-art sparse, optimisation-based VI-SLAM system. It receives stereo greyscale image pairs and inertial measurements and produces state estimates at IMU rate which include the full MAV state and IMU biases. In the case of MAVs equipped with LiDARs, we also integrate our previous work [Boche et al., 2024], which formulates residuals based on occupancy maps and their gradients to optimise for consistency between (a) incoming measurements and previous submaps, and (b) overlapping submaps. The reader is referred to [Boche et al., 2024] for details.

6.4.2 Occupancy Mapping

We use *supereight2* [Funk et al., 2021] for volumetric occupancy mapping using octrees. *Supereight2* explicitly represents *free* space and propagates minimum and maximum occupancy values to the octree root, allowing safe and efficient path planning. The original *supereight2* only allows integrating depth images, either from a depth camera or by projecting a structured LiDAR scan. In order to support dynamically moving LiDAR sensors, we adopt the integration scheme from [Boche et al., 2024].

Supereight2 forms the basis of our submapping framework, as in [Boche et al., 2024] and [Barbas Laina et al., 2024]. New submaps are generated based on the geometric overlap in the case of LiDAR sensors, as in [Boche et al., 2024], or based on visual keyframes in the case of depth cameras, as in [Barbas Laina et al., 2024]. In both cases submaps are anchored to *OKVIS2* keyframe states and are transformed along with them on pose optimisation, including loop closures. Once a new submap is created, the previous one is frozen, allowing only rigid transformations for the remainder of the mission.

As shown in the real-world experiment in [Barbas Laina et al., 2024], the level of drift of *OKVIS2* is less than 0.5 %. However, even less than 0.5 m of drift after travelling 100 m can be enough to cause a crash if a monolithic map is used, necessitating the use of submaps for large-scale navigation and exploration.

The memory usage of the submapping system is higher than using a monolithic map, mainly due to overlaps between the submaps. These overlaps are necessary for navigation since they allow some amount of relative movement between the submaps without introducing unmapped regions between them. Given the fact that only the currently active submap is ever updated, the CPU time required for submapping is no greater than that required for monolithic mapping and in fact it can even be lower. Due to the limited spatial extent of submaps, they can reach the same spatial resolution as a larger monolithic map using fewer octree levels. This results in faster data fusion and access since fewer tree levels need to be traversed.

6.4.3 Exploration Planning

We extend the exploration planner for monolithic maps we proposed in Chapter 4 to account for submaps. The key ideas of this planner remain unchanged and are

re-stated here for convenience. Candidate next positions $\mathbf{r}_j \in V$, $j \in \{1 \dots n_c\}$, $n_c \in \mathbb{N}^+$ are sampled close to frontiers, since they correspond to regions that will expand the map if observed. A path is planned from the current MAV position to each candidate and its estimated duration t_j is computed assuming the MAV flies at $v_{\max}$ and $\omega_{\max}$. A 360° map entropy raycast is performed from each candidate position $\mathbf{r}_j$ producing a gain image. Given the sensor's horizontal field of view α_h , a sliding window optimisation is performed on the gain image to determine the yaw angle ψ_j resulting in the highest potential gain g_j . The utility of each candidate j is computed as $u_j = g_j / t_j$, essentially maximising information gain over time. The candidate with the highest utility becomes the next goal and the process is repeated once the MAV reaches it.

The exploration planner presented in this chapter differs from the one presented in Chapter 4 in two aspects. First, we compute global frontiers from the collection of submaps $\mathcal{M}$, presented in Section 6.5 as one of our core contributions. Second, the yaw optimisation over the gain image is performed differently for sensors with $\alpha_h = 360^\circ$, which the method of Chapter 4 is not designed to handle. In this case the candidate yaw angle ψ_j can be chosen arbitrarily while the candidate gain g_j is just computed over the whole image. For simplicity, we compute ψ_j using a sliding window corresponding to a horizontal field of view smaller than 360° .

6.4.4 MAV Controller

We use a linear MPC based on [Tzoumanikas et al., 2019] for MAV trajectory following, with modifications as described in [Barbas Laina et al., 2024]. In short, the MPC is modified to ensure correct trajectory tracking even in the case of major odometry changes, such as loop closures. This is achieved by anchoring trajectories to OKVIS2 keyframe states and elastically deforming them as the keyframe states are updated over time.

6.5 Global Frontier Computation

Candidate next positions are sampled exclusively close to frontiers because frontiers correspond to regions that will expand the mapped region if observed. In the case of multiple submaps, a set of global frontiers must be computed that consid-

ers all submaps and the overlapping regions between them. This means that if a frontier in one submap corresponds to a fully observed region in another submap then it should not be considered a global frontier.

To achieve efficient global frontier computation we separate the process in two stages. First, each time a submap is updated, we update its local frontiers. This means frontiers are computed while considering only this submap in isolation. Once a submap is frozen due to a new submap being created, its set of local frontiers is also frozen. Then, global frontiers are computed taking into account all currently known local frontiers each time a new iteration of the exploration planner is initiated. Global frontiers are only used for sampling candidate next views so this scheme ensures they are only computed when necessary. The following describes the local and global frontier computation in detail.

6.5.1 Local Frontier Computation

For each submap M_i we keep track of a set of local frontiers $\mathcal{F}_i^k$ at time-step k , by considering M_i in isolation from other submaps. Local frontiers are *free* voxels that (i) have at least one *unknown* face neighbour voxel or (ii) are at the submap boundary. The latter case is essential for correct global frontiers when using bounded-extent submaps.

Local submap frontiers $\mathcal{F}_i^k$ at time-step k are updated in two steps. First, all preexisting local frontiers $\mathcal{F}_i^{k-1}$ are re-tested so that the out-of-date frontiers $\mathcal{F}_{i,\text{stale}}^{k-1}$ are detected. Then, the set of new frontiers $\mathcal{F}_{i,\text{new}}^k$ is computed by only testing voxels that were modified since $\mathcal{F}_i^{k-1}$ was computed. Finally, the local frontiers at time-step k are computed as

$$\mathcal{F}_i^k = (\mathcal{F}_i^{k-1} \setminus \mathcal{F}_{i,\text{stale}}^{k-1}) \cup \mathcal{F}_{i,\text{new}}^k. \quad (6.1)$$

The detection of stale frontiers $\mathcal{F}_{i,\text{stale}}^{k-1}$ can be performed in parallel with the computation of the new frontiers $\mathcal{F}_{i,\text{new}}^k$, taking advantage of multi-core processors.

6.5.2 Global Frontier Computation

The global frontiers $\mathcal{F}_g^k$ at time-step k are computed as

$$\mathcal{F}_g^k = \bigcup_{i=1}^{n_m} \mathcal{F}_{g,i}^k, \quad (6.2)$$

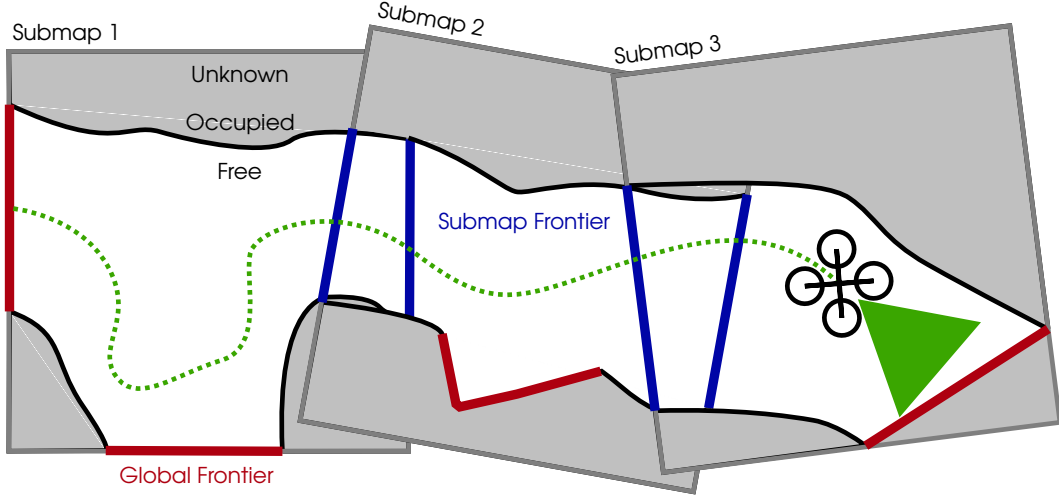


Figure 6.3: Local-only (blue) and global (red) frontiers in a set of submaps. The local-only submap frontiers correspond to regions mapped in other submaps and are thus not considered to be global frontiers.

where $\mathcal{F}_{g,i}^k \subseteq \mathcal{F}_i$ are the local frontiers of M_i that are also global frontiers at time-step k , and $\mathcal{F}_i$ are the last-known frontiers of M_i . The set $\mathcal{F}_{g,i}^k$ is computed by testing each frontier in $\mathcal{F}_i$ against all other submaps

$$\mathcal{F}_{g,i}^k = \left\{ f_i \in \mathcal{F}_i, \bigwedge_{l \neq i} h(f_i, M_l) \right\}, \quad (6.3)$$

where the Boolean-valued function $h(f_i, M_l)$ indicates whether frontier f_i of M_i is also a frontier when considering M_l . $h(f_i, M_l)$ is *true* iff $f_i \notin M_l$ or f_i corresponds to an *unknown* region of M_l or f_i corresponds to a local frontier of M_l . The computation of $\mathcal{F}_{g,i}^k$ can be performed in parallel for each submap, taking advantage of multi-core processors.

Figure 6.3 shows an example of local and global frontiers.

6.6 Evaluation

6.6.1 Simulation Results

Our simulation setup consists of *Gazebo* [Open Robotics, 2021] and the *PX4* autopilot [PX4, 2012] simulated in software, mimicking the setup of the real-world MAV. The simulated MAV is based on the RMF-Owl [De Petris et al., 2022], a $0.38 \times 0.38 \times 0.24$ m quadcopter equipped with a LiDAR sensor and an Intel Re-

alSense D455 RGB-D camera. The evaluation is done in the warehouse-like environment *Depot*, shown in Figure 6.4. All simulated experiments were conducted on a computer with an Intel Core i7-1165G7 CPU and 32 GB of RAM.

We compare our method with the state-of-the-art, submap-based, 3D exploration planner *GLocal* [Schmid et al., 2021], in terms of exploration speed, environment reconstruction accuracy and completeness, as well as CPU and memory usage. *GLocal* requires a LiDAR sensor and performs submap alignment using *Voxgraph* [Reijgwart et al., 2019], similarly to our approach based on [Boche et al., 2024], and is described in more detail in Section 2.5.4. Thus, we only evaluate LiDAR-based exploration in simulation. Both approaches receive poses from *OKVIS2*, including visual loop closures. Our approach includes submap alignment based on [Boche et al., 2024], whereas *GLocal* uses map-to-map alignment from *Voxgraph*. We use the MAV controller described in Section 6.4.4 for both pipelines. We made an effort to use the same parameters for both methods where applicable, the most important of which are listed in Table 6.1.

Parameter		Value
Map resolution		0.1 m
MAV radius	R	0.5 m
MAV maximum linear velocity	$v_{\max}$	0.5 m/s
MAV maximum angular velocity	$\omega_{\max}$	0.5 rad/s
LiDAR resolution		360×180
LiDAR horizontal/vertical FoV	α_h, α_v	$360^\circ, 90^\circ$
LiDAR measurement range	$[d_{\min}, d_{\max}]$	[1, 10] m
Candidate poses sampled (n/a to <i>GLocal</i>)	n_c	20

Table 6.1: Simulated experiment parameters.

Explored Volume

In order to avoid discrepancies in the explored volume due to different mapping frameworks, we record the LiDAR measurements and corresponding ground-truth MAV poses. We use them to construct a monolithic *supereight2* map, measuring the explored volume after each integration. For each method, we performed 10 runs using poses from *OKVIS2* and another 10 runs using ground-truth poses as a baseline. The median, 10th and 90th percentiles of the percentage of the total volume explored by the two methods is shown in Figure 6.5. Our method outper-

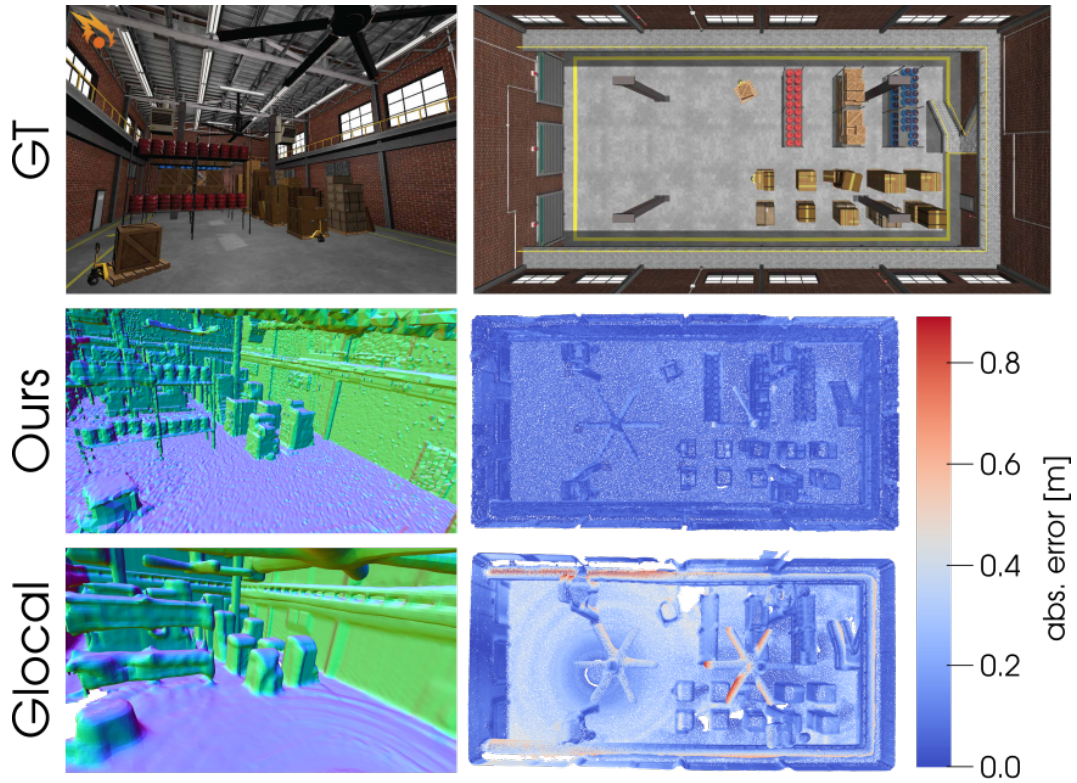


Figure 6.4: Perspective and top-down visualisation of the *Depot* environment. [Top] Ground-truth. [Middle] and [Bottom] Final reconstruction (left) and mesh accuracy (right) using our approach and *GLocal* respectively.

forms *GLocal* in terms of exploration speed, consistency and final environment coverage. While our approach nearly always explores 100% of the volume within 300 s, *GLocal* sometimes fails to explore the whole volume. Using ground-truth poses results in less variance between runs, which can be attributed to the more consistent maps resulting in fewer frontiers and more free space for the MAV to navigate through.

Safety Evaluation

We evaluate the safety of the exploration planner by computing the minimum distance between the *Depot* ground-truth mesh and every point of the ground-truth MAV trajectory. Figure 6.6 shows a histogram of these distances across all 10 missions. Our approach in general succeeds to navigate safely even in proximity to obstacles while *GLocal* violates the safety radius $R = 0.5$ m significantly more often, leading to a higher risk of collisions.

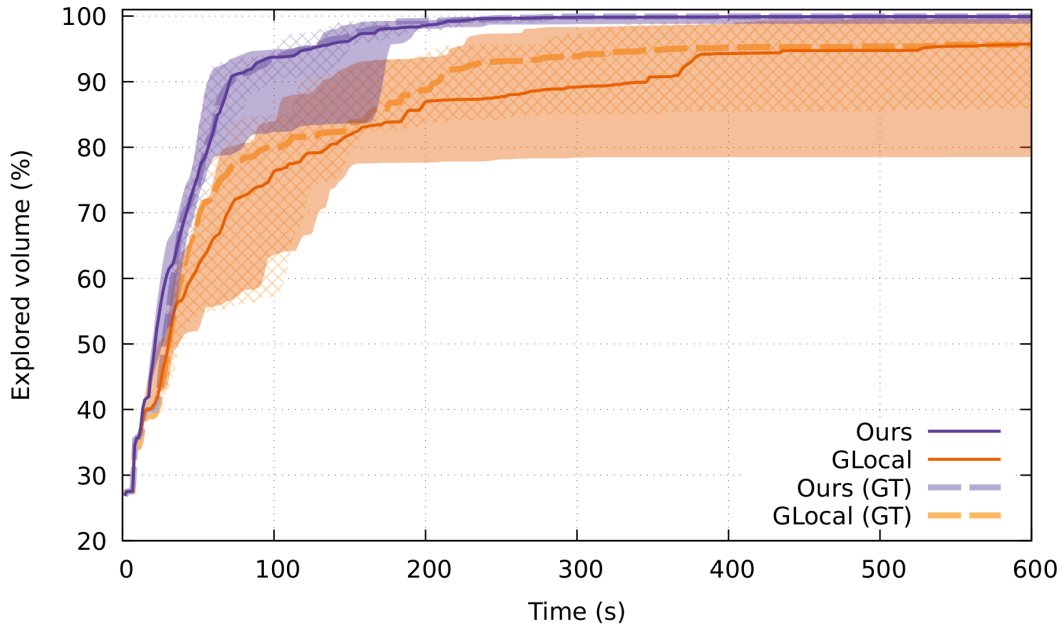


Figure 6.5: Explored volume median, 10th and 90th percentiles for 10 runs in the *Depot* environment. Solid for runs using SLAM poses and dashed/hatched for runs using ground-truth poses

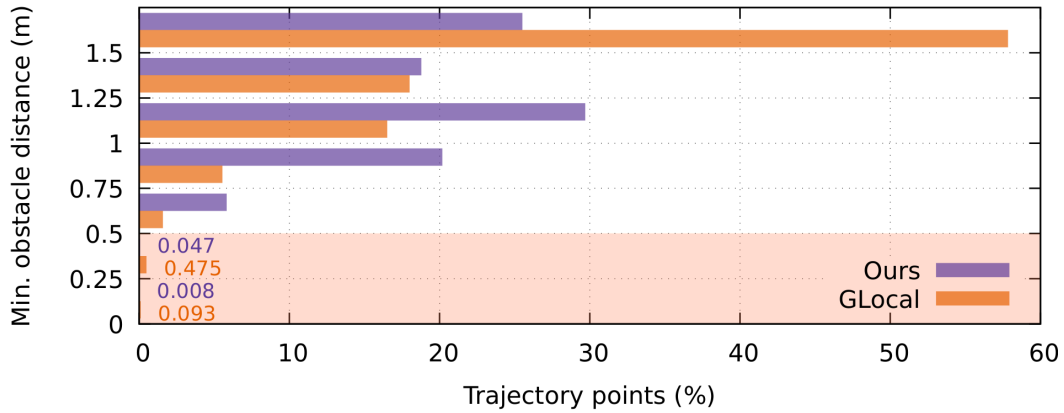


Figure 6.6: Histogram of minimum distance of the MAV to obstacles across 10 simulated missions with SLAM poses and a safety radius of 0.5 m. The bin size is 0.25 m.

Reconstruction Quality

We also evaluate the quality of the final mesh reconstructions at a 0.1 m resolution obtained from a post-processing step. In our method, the final map reconstruction is computed using poses estimated after a final bundle adjustment to provide the best possible accuracy. We use the ground-truth *Depot* mesh to compute the Root

Mean Squared Error (RMSE) of the reconstruction. The completeness is computed as the percentage of the reconstruction within 0.2 m or 0.4 m of the ground truth. The mean RMSE and completeness of all 10 runs are presented in Table 6.2. Our approach yields significantly better results in both metrics, showcasing it is possible to achieve safe path planning without sacrificing reconstruction quality. This is explained to some extent by the fact that the *Voxblox* mapping framework used by *GLocal* artificially inflates obstacles for safer path planning. A visual comparison of the final reconstructions as well as their accuracy is presented in Figure 6.4.

	RMSE (m) ↓	Completeness (%) ↑	
		within 0.2 m	within 0.4 m
Ours	0.112	86.07	97.73
GLocal	0.219	44.19	76.97
Ours (GT)	0.095	89.89	97.98
GLocal (GT)	0.213	45.29	78.20

Table 6.2: Mesh reconstruction RMSE and completeness using both SLAM and ground-truth (GT) poses in the *Depot* environment.

Resource Usage

We also compare the computational resources required by our method and *GLocal* in Table 6.3. Even though both methods compute frontiers only as needed, ours benefits from the more efficient map representation of *supereight2*. While we only compute the utility for a limited number of candidate next positions, in *GLocal* it is computed for each vertex of the tree used for local planning, requiring significantly more time. The planning time includes utility computation, exploration planning, and path planning. The high planning time in *GLocal* is due to a dedicated thread re-planning at a high rate, accumulating a large amount of CPU time, even after ignoring trivial planning iterations requiring less than 200 ms. Finally, the smaller memory usage of our system is due to the efficiency of *supereight2* maps.

	Frontiers (s)	Utility (s)	Planning (s)	Memory (GB)
Ours	16 ± 7	11 ± 6	134 ± 45	6.7 ± 1.0
GLocal	52 ± 7	275 ± 84	421 ± 76	23.5 ± 1.8

Table 6.3: Mean and standard deviation of per-mission frontier, utility, and path and exploration planning computation time and memory usage.

6.6.2 Real World Experiments

Real-world experiments were conducted to further showcase the effectiveness of the proposed exploration approach on resource-constrained platforms and demonstrate its applicability to both LiDAR sensors and depth cameras. The experiments were carried out in an $8.0 \times 8.1 \times 3.7 \text{ m} \approx 240 \text{ m}^3$ room containing some tall obstacles.



Figure 6.7: MAVs used for real-world experiments. [Left] Leica BLK2Fly with LiDAR. [Right] Custom MAV with Intel RealSense D455 RGB-D camera.

For LiDAR exploration, we use a Leica BLK2Fly MAV, equipped with 5 cameras, 5 IMUs and a single-beam, dual-axis spinning LiDAR sensor. For VI-SLAM, only the front, left, and bottom cameras and the bottom IMU are used. The LiDAR has an effective frequency of 5 Hz and a 360° field of view in both axes, although the MAV body occludes part of it, resulting in α_h and α_v both being less than 360° . On-board filtering of the LiDAR point clouds reduces the number of points to $\approx 100,000$ to $200,000$ points per second. As on-board resources are not made available for general use by the manufacturer, the sensor data is streamed via WiFi to a laptop where LVI-SLAM, mapping and exploration planning are running. The laptop used for off-board processing has an Intel Core i7-13850HX CPU and 32 GB of RAM.

For depth camera exploration we use a custom-built quadcopter based on the Holybro S500 frame and equipped with an Intel RealSense D455 stereo RGB-D camera and an NVIDIA Jetson Orin NX 16 GB on-board computer. All computations, including VI-SLAM, dense occupancy mapping, exploration planning and MPC are performed on-board.

Both MAV platforms are shown in Figure 6.7. The parameters used for the experiments are listed in Table 6.4.

We conducted one experiment using each MAV platform. The final mesh reconstructions and estimated MAV trajectories are shown in Figure 6.1 while Fig-

		LiDAR	Depth camera
Map resolution		0.1 m	0.1 m
MAV radius	R	0.6 m	0.6 m
MAV maximum linear velocity	$v_{\max}$	1.0 m/s	0.5 m/s
MAV maximum angular velocity	$\omega_{\max}$	-	0.785 rad/s
Sensor measurement range	$[d_{\min}, d_{\max}]$	$[0.25, 10]$ m	$[0.2, 4]$ m

Table 6.4: Real-world experiment parameters.

ure 6.8 shows the observed volume over time. As expected, due to its much larger field of view, the LiDAR-equipped MAV achieves a higher exploration speed and reaches an almost complete environment coverage within less than a minute. Nevertheless, even using a depth camera with a much more limited field of view, we can almost fully explore the room in an efficient manner.

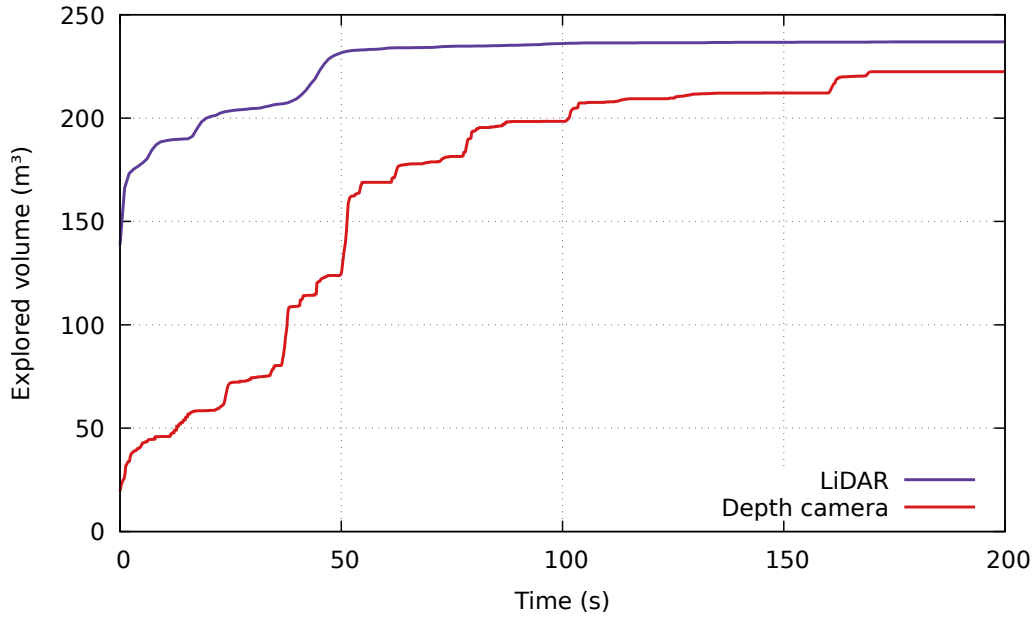


Figure 6.8: Explored volume over time for the two real-world experiments.

Submapping Results

It is interesting to compare the submap generation rate between the two experiments given their different sensor modalities and submap generation strategies. In the case of the high-Field of View (FoV) LiDAR, new submaps are generated based on the geometric overlap between new measurements and the existing submaps.

In the case of the lower-FoV depth camera, new submaps are generated every 3 visual keyframes. Table 6.5 contains the number of submaps generated over time and over distance for both experiments. Given its sensor’s lower FoV, the MAV equipped with a depth camera has to yaw more often compared to the LiDAR-equipped one in order to observe the whole environment. This results in frequent visual keyframes which in turn result in many new submaps being generated. On the other hand, generating submaps based on geometric overlap, combined with the high range and FoV of the LiDAR, requires large MAV movements to trigger the creation of a new submap. While using a geometric overlap metric for submap creation in the depth camera case will reduce the number of submaps, it is not an obvious improvement in the overall system. The higher keyframe rate of the depth camera pipeline will result in larger submaps being separated by more keyframes, which may violate the assumption that drift within a submap is negligible.

	LiDAR	Depth camera
Submaps	8	45
Experiment duration	180 s	185 s
Trajectory length	19 m	55 m
Submaps over time	0.04 submaps/s	0.28 submaps/s
Submaps over distance	0.42 submaps/m	0.79 submaps/m

Table 6.5: Real-world experiment results.

6.7 Conclusion

In this chapter, we described an efficient and robust submap-based autonomous exploration method that we demonstrated to accept both depth cameras and LiDARs as input modalities. Our method leverages submapping and (LiDAR)-VI-SLAM, in order to achieve accurate and consistent environment mapping despite odometry drift, thus ensuring safe operation. The efficient computation of global frontiers from the aggregated submaps allows us to apply only one unified global planning approach, rendering the distinction between local and global planning that state-of-the-art methods employ unnecessary. Compared to a state-of-the-art submap-based large-scale exploration framework, our method achieves faster exploration and more accurate environment reconstructions. It is further deployed on two real-world MAV platforms, one using a depth camera, and the other one a LiDAR.

Conclusion

7.1 Summary of Contributions

The contributions of this thesis are focused on two themes: i) efficient autonomous exploration algorithms suitable for MAVs, ii) extending exploration from a purely geometric formulation to an object-centric one.

Chapter 4 presents a fast and efficient hybrid exploration method. It combines aspects of frontier-based, information-theoretic, and sampling-based exploration methods. Candidate next views are sampled close to frontiers which is much more sample-efficient than sampling in the entire *free* space. Computationally expensive frontier clustering in 3D is avoided by exploiting the hierarchical nature of the underlying octree-based occupancy map, resulting in fast, incremental frontier computation. Candidate next views are ranked using a utility that expresses information gain over time, resulting in a planner geared towards maximising the map coverage and quality over time. The method is evaluated in several simulated experiments and is deployed on-board a real-world an MAV.

Chapter 5 describes an approach that detects and reconstructs objects of interest while exploring and mapping the environment. The exploration planner presented in Chapter 4 is extended by incorporating more gains in the candidate view utility and modifying the sampling strategy. In order for an object of interest to be detected by the semantic instance segmentation component it must appear large enough in the captured colour images. To this end, a background distance gain is added to the utility function that rewards observing surfaces from a small enough distance. Another goal of this approach is creating high-quality reconstructions of

the objects of interest, which, given the fact that the measurement noise of depth cameras increases with distance, requires close-up observations of objects. This is achieved by adding an additional gain component to the utility function so that observations of known objects from a small distance are rewarding and by sampling candidate views close to objects in addition to close to frontiers. The proposed approach is evaluated extensively in simulation, showing improved object reconstruction accuracy without sacrificing exploration speed. It is also evaluated on a real-world MAV, with mapping, exploration planning and object detection all running on-board.

Chapter 6 introduces a system suitable for fast and efficient autonomous exploration of large-scale environments without the need for any prior infrastructure. To allow handling odometry drift, the environment map consists of a series of occupancy submaps anchored to SLAM keyframe states, allowing the relative submap poses to be updated upon better state estimates. Unlike previous systems which use hierarchical exploration planners, we propose an efficient, incremental submap-local and global frontier update method that allows using a single, global exploration planner, similar to the one presented in Chapter 4. The proposed system is compared to a state-of-the-art, submap-based exploration method in simulation and it achieves faster exploration, more accurate reconstruction and safer navigation. The system is suitable for both depth cameras and LiDAR sensors, as shown by experiments on two different real-world MAVs.

7.2 Limitations and Future Work

The methods proposed in this thesis are not without limitations and there is room for further research in a number of directions.

7.2.1 Feasible and Safe MAV Trajectories

The path generation method used in Chapters 4 to 6 does not take the MAV kinodynamic constraints into account, which can result in non-feasible trajectories. Tracking such a trajectory poses a collision risk if no further measures are taken. As such, the safety of the proposed methods can be greatly increased by integrating trajectory planning that takes the MAV kinodynamics into account. Another

layer of safety can be added in the form of safe control, for example filtering the reference state to avoid collisions as in [Zhou et al., 2024].

7.2.2 Object Detection, Matching and Reconstruction

Recent advances in Vision-Language Models (VLMs) and foundation models have allowed reliable detection of objects beyond a closed set of classes, as is the case for the method described in Chapter 5. Employing these new models could potentially allow generating semantically-rich maps that support a larger variety of queries. This could open the door for more flexible exploration systems that can adapt to different tasks and exploration planners that can make use of semantic information in the local and global scale.

The object matching in Chapter 5 is purely image-based, which can result in both over- and under-segmentation of objects. A matching algorithm that also takes 3D and semantic information into account would improve the robustness of object matching and thus the quality of object reconstructions.

All the methods presented in Chapters 4 to 6 assume that the environment is static and dynamic objects will cause artefacts in the map. There are existing systems that are able to track dynamic objects over a short [Xu et al., 2019] or long [Schmid et al., 2024] time-frame. However doing both at the same time while consistently matching instances over long time periods remains challenging, even more so when collision free trajectories must be computed at a high rate.

7.2.3 Robust, Generic Exploration

While there are a large number of autonomous exploration methods designed for different types of environments, there is still no single method that excels in all environment types. The approaches presented in Chapters 4 to 6 are better suited to exploration of interior spaces, where frontiers provide a useful indication of where to go next. Frontiers are less useful for guiding the exploration in outdoor environments, for example in a forest, since there will be a large number of frontiers in all directions. One approach would be a high-level exploration planner that switches between sub-planners based on the structure and characteristics of the immediate environment.

7.2.4 Multi-robot Exploration

Scaling a system to multiple collaborating robots is a challenge and autonomous exploration is no exception. It will allow exploring and mapping larger environments than any single robot can, especially important in the case of MAVs which have a limited battery life. Open issues include computationally efficient task allocation to the individual robots, exchanging information on the exploration progress to perform more informed future actions and avoid redundant work, and achieving a consistent global map, necessary for some downstream tasks, especially in the absence of GPS.

7.2.5 Autonomous Robots in the Future

A bigger question is what the role of autonomous robots will be in a world with widening wealth inequality and that is only starting to be affected by climate change. Will they be used to help battle climate change, to protect people and ecosystems from its worst effects? Or will they be used to disproportionately benefit the wealthy, at the detriment of the majority of the population and the environment? This is a social issue rather than a technical one but I think an important one to consider. As robotics researchers we can influence the way our work is used, select what research directions to pursue and who to work for. I am hopeful that the majority of researchers, knowing the impact of their work on the world, will choose to benefit the many.

Bibliography

- [Abdulla, 2017] Abdulla, W. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. https://github.com/matterport/Mask_RCNN. 98
- [Achiam et al., 2023] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). GPT-4 technical report. *arXiv preprint arXiv:2303.08774*. 34
- [Adamkiewicz et al., 2022] Adamkiewicz, M., Chen, T., Caccavale, A., Gardner, R., Culbertson, P., Bohg, J., and Schwager, M. (2022). Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2):4606–4613. 30
- [Agha et al., 2021] Agha, A., Otsu, K., Morrell, B., Fan, D. D., Thakker, R., Santamaria-Navarro, A., Kim, S.-K., Bouman, A., Lei, X., Edlund, J., et al. (2021). NeBula: Quest for robotic autonomy in challenging environments; Team CoSTAR at the DARPA subterranean challenge. *arXiv preprint arXiv:2103.11470*. 83
- [Akbari and Bernardini, 2021] Akbari, A. and Bernardini, S. (2021). Informed autonomous exploration of subterranean environments. *IEEE Robotics and Automation Letters*, 6(4):7957–7964. 45
- [Alexis et al., 2012] Alexis, K., Nikolakopoulos, G., and Tzes, A. (2012). Model predictive quadrotor control: Attitude, altitude and position experimental studies. *IET Control Theory & Applications*, 6(12):1812–1827. 36
- [Armeni et al., 2019] Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J., and Savarese, S. (2019). 3D scene graph: A structure for unified semantics, 3D space, and camera. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 32

- [Asgharivaskasi and Atanasov, 2021] Asgharivaskasi, A. and Atanasov, N. (2021). Active Bayesian multi-class mapping from range and semantic segmentation observations. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 48, 85
- [Ashour et al., 2020] Ashour, R., Taha, T., Dias, J. M. M., Seneviratne, L., and Almoosa, N. (2020). Exploration for object mapping guided by environmental semantics using UAVs. *Remote Sensing*, 12(5). 48, 85
- [Baca et al., 2016] Baca, T., Loianno, G., and Saska, M. (2016). Embedded model predictive control of unmanned micro aerial vehicles. In *21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 992–997. IEEE. 36
- [Bai et al., 2016] Bai, S., Wang, J., Chen, F., and Englot, B. (2016). Information-theoretic exploration with Bayesian optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1816–1822. IEEE. 42
- [Barbas Laina et al., 2024] Barbas Laina, S., Boche, S., Papatheodorou, S., Tzoumanikas, D., Schaefer, S., Chen, H., and Leutenegger, S. (2024). Scalable autonomous drone flight in the forest with visual-inertial SLAM and dense submaps built without LiDAR. *arXiv preprint arXiv:2403.09596*. 60, 61, 106, 107
- [Best et al., 2022] Best, G., Garg, R., Keller, J., Hollinger, G., and Scherer, S. (2022). Resilient Multi-Sensor Exploration of Multifarious Environments with a Team of Aerial Robots. In *Robotics: Science and Systems (RSS)*, New York City, NY, USA. 84
- [Bircher et al., 2016] Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding horizon next-best-view planner for 3D exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1462–1468. IEEE. 43, 44, 67, 75
- [Bircher et al., 2018] Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2018). Receding horizon path planning for 3D exploration and surface inspection. *Autonomous Robots*, 42(2):291–306. 44, 72, 76, 80, 83, 101
- [Bissmarck et al., 2015] Bissmarck, F., Svensson, M., and Tolt, G. (2015). Efficient algorithms for next best view evaluation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5876–5883. 67
- [Bloesch et al., 2017] Bloesch, M., Burri, M., Omari, S., Hutter, M., and Siegwart, R. (2017). Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072. 38

- [Boche et al., 2024] Boche, S., Barbas Laina, S., and Leutenegger, S. (2024). Tightly-coupled LiDAR-visual-inertial SLAM and large-scale volumetric occupancy mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan. IEEE. 39, 60, 101, 103, 105, 106, 110
- [Bolya et al., 2019] Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019). YOLACT: Real-time instance segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9157–9166, Seoul, Korea. IEEE. 89, 98
- [Border et al., 2024] Border, R., Chebrolu, N., Tao, Y., Gammell, J. D., and Fallon, M. (2024). Osprey: Multi-session autonomous aerial mapping with LiDAR-based SLAM and next best view planning. *IEEE Transactions on Field Robotics*. 47
- [Bouabdallah, 2007] Bouabdallah, S. (2007). *Design and control of quadrotors with application to autonomous flying*. PhD thesis, École Polytechnique Fédérale de Lausanne. 36
- [Bourgault et al., 2002] Bourgault, F., Makarenko, A. A., Williams, S. B., Grocholsky, B., and Durrant-Whyte, H. F. (2002). Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 540–545. IEEE. 42, 45
- [Brescianini and D’Andrea, 2016] Brescianini, D. and D’Andrea, R. (2016). Design, modeling and control of an omni-directional aerial vehicle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3261–3266. IEEE. 36
- [Campos et al., 2021] Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. (2021). ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890. 38, 101
- [Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660. 33
- [Carrillo et al., 2017] Carrillo, H., Dames, P., Kumar, V., and Castellanos, J. (2017). Autonomous robotic exploration using a utility function based on Rényi’s general theory of entropy. *Autonomous Robots*, 42. 67
- [Castillo et al., 2003] Castillo, P., Dzul, A., and Lozano, R. (2003). Real-time stabilization and tracking of a four rotor mini-rotorcraft. In *European Control Conference (ECC)*, pages 3123–3128. IEEE. 36

- [Chang et al., 2017] Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV)*, pages 667–676, Qingdao, China. 85, 95, 142
- [Chaplot et al., 2020] Chaplot, D. S., Gandhi, D. P., Gupta, A., and Salakhutdinov, R. R. (2020). Object goal navigation using goal-oriented semantic exploration. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 4247–4258. 48, 83, 85
- [Charrow et al., 2015a] Charrow, B., Kahn, G., Patil, S., Liu, S., Goldberg, K., Abbeel, P., Michael, N., and Kumar, V. (2015a). Information-theoretic planning with trajectory optimization for dense 3D mapping. In *Robotics: Science and Systems (RSS)*, pages 3–12, Rome, Italy. 45
- [Charrow et al., 2015b] Charrow, B., Liu, S., Kumar, V., and Michael, N. (2015b). Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4791–4798, Seattle, WA, USA. IEEE. 45, 67, 71
- [Chen et al., 2016] Chen, J., Liu, T., and Shen, S. (2016). Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1476–1483. IEEE. 36
- [Chen et al., 2024] Chen, L., Chen, W., Wang, R., and Pollefeys, M. (2024). Leveraging neural radiance fields for uncertainty-aware visual localization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6298–6305. IEEE. 30
- [Chen et al., 2025] Chen, T., Shorinwa, O., Bruno, J., Swann, A., Yu, J., Zeng, W., Nagami, K., Dames, P., and Schwager, M. (2025). Splat-Nav: Safe real-time robot navigation in Gaussian splatting maps. *IEEE Transactions on Robotics*, pages 1–20. 31
- [Cieslewski et al., 2017] Cieslewski, T., Kaufmann, E., and Scaramuzza, D. (2017). Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2135–2142, Vancouver, BC, Canada. IEEE. 41, 66, 78
- [Connolly, 1985] Connolly, C. I. (1985). The determination of next best views. *IEEE International Conference on Robotics and Automation (ICRA)*, 2:432–435. 43, 67
- [Dai et al., 2020] Dai, A., Papatheodorou, S., Funk, N., Tzoumanikas, D., and Leutenegger, S. (2020). Fast frontier-based information-driven autonomous exploration with an MAV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9570–9576, Paris, France. 17, 65, 83, 101

- [Dang et al., 2019] Dang, T., Khattak, S., Mascarich, F., and Alexis, K. (2019). Explore locally, plan globally: A path planning framework for autonomous robotic exploration in subterranean environments. In *19th International Conference on Advanced Robotics (ICAR)*, pages 9–16. IEEE. 47, 102, 103
- [Dang et al., 2018a] Dang, T., Papachristos, C., and Alexis, K. (2018a). Autonomous exploration and simultaneous object search using aerial robots. In *IEEE Aerospace Conference*, pages 1–7, Big Sky, MT, USA. IEEE. 48, 83, 85
- [Dang et al., 2018b] Dang, T., Papachristos, C., and Alexis, K. (2018b). Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2526–2533. IEEE. 67
- [DARPA, 2021] DARPA (2017–2021). SubT: DARPA subterranean challenge. <https://www.darpa.mil/research/programs/darpa-subterranean-challenge>. 83
- [De Petris et al., 2022] De Petris, P., Nguyen, H., Dharmadhikari, M., Kulkarni, M., Khedekar, N., Mascarich, F., and Alexis, K. (2022). RMF-Owl: A collision-tolerant flying robot for autonomous subterranean exploration. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 536–543. IEEE. 109
- [Delmerico et al., 2018] Delmerico, J., Isler, S., Sabzevari, R., and Scaramuzza, D. (2018). A comparison of volumetric information gain metrics for active 3D object reconstruction. *Autonomous Robots*, 42(2):197–208. 67
- [Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271. 41
- [Dolgov et al., 2010] Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2010). Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501. 35
- [Duberg and Jensfelt, 2020] Duberg, D. and Jensfelt, P. (2020). UFOMap: An efficient probabilistic 3D mapping framework that embraces the unknown. *IEEE Robotics and Automation Letters*, 5(4):6411–6418. 25
- [Duberg and Jensfelt, 2022] Duberg, D. and Jensfelt, P. (2022). UFOExplorer: Fast and scalable sampling-based exploration with a graph-based planning structure. *IEEE Robotics and Automation Letters*, 7(2):2487–2494. 44, 83
- [Falanga et al., 2018] Falanga, D., Kleber, K., Mintchev, S., Floreano, D., and Scaramuzza, D. (2018). The foldable drone: A morphing quadrotor that can squeeze and fly. *IEEE Robotics and Automation Letters*, 4(2):209–216. 36

- [Faria et al., 2019] Faria, M., Maza, I., and Viguria, A. (2019). Applying frontier cells based exploration and Lazy Theta* path planning over single grid-based world representation for autonomous inspection of large 3D structures with an UAS. *Journal of Intelligent & Robotic Systems*, 93(1):113–133. 42, 66
- [Fisher et al., 2021] Fisher, A., Cannizzaro, R., Cochrane, M., Nagahawatte, C., and Palmer, J. L. (2021). ColMap: A memory-efficient occupancy grid mapping framework. *Robotics and Autonomous Systems*, 142:103755. 26
- [Fridovich-Keil et al., 2022] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. (2022). Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5501–5510. 30, 31
- [Funk, 2023] Funk, N. (2023). *Multi-resolution mapping and planning for UAV navigation in attitude constrained environments*. PhD thesis, Imperial College London, London, United Kingdom. 54, 55, 56
- [Funk et al., 2023] Funk, N., Tarrio, J., Papatheodorou, S., Alcantarilla, P. F., and Leutenegger, S. (2023). Orientation-aware hierarchical, adaptive-resolution A* algorithm for UAV trajectory planning. *IEEE Robotics and Automation Letters*, 8(10):6723–6730. 35
- [Funk et al., 2021] Funk, N., Tarrio, J., Papatheodorou, S., Popović, M., Alcantarilla, P. F., and Leutenegger, S. (2021). Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning. *IEEE Robotics and Automation Letters*, 6(2):3553–3560. 26, 39, 57, 58, 87, 106
- [Furrer et al., 2016] Furrer, F., Burri, M., Achtelik, M., and Siegwart, R. (2016). *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer. 75
- [Gammell et al., 2014] Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 35, 63, 72, 91
- [Gao et al., 2020] Gao, F., Wang, L., Zhou, B., Zhou, X., Pan, J., and Shen, S. (2020). Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments. *IEEE Transactions on Robotics*, 36(5):1526–1545. 36
- [Gao et al., 2018a] Gao, F., Wu, W., Lin, Y., and Shen, S. (2018a). Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 344–351. IEEE. 34

- [Gao et al., 2018b] Gao, W., Booker, M., Adiwahono, A. H., Yuan, M., Wang, J., and Yau, W.-Y. (2018b). An improved frontier-based approach for autonomous exploration. *15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 292–297. 41, 66
- [Geiger et al., 2012a] Geiger, A., Lenz, P., and Urtasun, R. (2012a). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 39
- [Geiger et al., 2012b] Geiger, A., Lenz, P., and Urtasun, R. (2012b). KITTI benchmark leaderboard, accessed on 2025-04-29. https://www.cvlibs.net/datasets/kitti/eval_odometry.php. 39
- [Geneva et al., 2020] Geneva, P., Eckenhoff, K., Lee, W., Yang, Y., and Huang, G. (2020). OpenVINS: A research platform for visual-inertial estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672. IEEE. 38
- [Grinvald et al., 2019] Grinvald, M., Furrer, F., Novkovic, T., Chung, J. J., Cadena, C., Siegwart, R., and Nieto, J. (2019). Volumetric instance-aware semantic mapping and 3D object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044. 32
- [Gu et al., 2024] Gu, Q., Kuwajerwala, A., Morin, S., Jatavallabhula, K. M., Sen, B., Agarwal, A., Rivera, C., Paul, W., Ellis, K., Chellappa, R., et al. (2024). Concept-Graphs: Open-vocabulary 3D scene graphs for perception and planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028, Yokohama, Japan. IEEE. 33
- [Guzhov et al., 2022] Guzhov, A., Raue, F., Hees, J., and Dengel, A. (2022). Audio-CLIP: Extending CLIP to image, text and audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE. 33
- [Han et al., 2019] Han, L., Gao, F., Zhou, B., and Shen, S. (2019). FIESTA: Fast incremental Euclidean distance fields for online motion planning of aerial robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4423–4430. IEEE. 28
- [Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107. 35, 43, 45, 47
- [He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE. 32, 89, 98

- [Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous robots*, 34:189–206. 25, 26, 57
- [Hudson et al., 2022] Hudson, N., Talbot, F., Cox, M., Williams, J., Hines, T., Pitt, A., Wood, B., Frousheger, D., Surdo, K. L., Molnar, T., et al. (2022). Heterogeneous ground and air platforms, homogeneous sensing: Team CSIRO Data61’s approach to the DARPA subterranean challenge. *Field Robotics*, 2:595–636. 83
- [Jatavallabhula et al., 2023] Jatavallabhula, K. M., Kuwajerwala, A., Gu, Q., Omama, M., Iyer, G., Saryazdi, S., Chen, T., Maalouf, A., Li, S., Keetha, N. V., Tewari, A., Tenenbaum, J., de Melo, C., Krishna, M., Paull, L., Shkurti, F., and Torralba, A. (2023). ConceptFusion: Open-set multimodal 3D mapping. In *Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea. 33
- [Jin et al., 2024a] Jin, L., Kuang, H., Pan, Y., Stachniss, C., and Popović, M. (2024a). STAIR: Semantic-targeted active implicit reconstruction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 395–402. 49
- [Jin et al., 2024b] Jin, R., Gao, Y., Wang, Y., Wu, Y., Lu, H., Xu, C., and Gao, F. (2024b). GS-Planner: A Gaussian-splatting-based planning framework for active high-fidelity reconstruction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11202–11209. IEEE. 45
- [Juliá et al., 2012] Juliá, M., Gil, A., and Reinoso, Ó. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33:427–444. 65
- [Kamel et al., 2018] Kamel, M., Verling, S., Elkhatib, O., Sprecher, C., Wulkop, P., Taylor, Z., Siegwart, R., and Gilitschenski, I. (2018). The Voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics & Automation Magazine*, 25(4):34–44. 36
- [Karaman and Frazzoli, 2011] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894. 63
- [Kassab et al., 2024] Kassab, C., Mattamala, M., Zhang, L., and Fallon, M. (2024). Language-EXtended Indoor SLAM (LEXIS): A versatile system for real-time visual scene understanding. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 15988–15994, Yokohama, Japan. IEEE. 34
- [Katragadda et al., 2024] Katragadda, S., Lee, W., Peng, Y., Geneva, P., Chen, C., Guo, C., Li, M., and Huang, G. (2024). NeRF-VINS: A real-time neural radiance field map-based visual-inertial navigation system. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10230–10237. IEEE. 30

- [Kaufman et al., 2018] Kaufman, E., Takami, K., Ai, Z., and Lee, T. (2018). Autonomous quadrotor 3D mapping and exploration using exact occupancy probabilities. In *Second IEEE International Conference on Robotic Computing (IRC)*, pages 49–55. IEEE. 42, 67
- [Kerbl et al., 2023] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):1–14. 30, 31
- [Kim et al., 2022] Kim, H., Kim, H., Lee, S., and Lee, H. (2022). Autonomous exploration in a cluttered environment for a mobile robot with 2D-map segmentation and object detection. *IEEE Robotics and Automation Letters*, 7(3):6343–6350. 50
- [Kirillov et al., 2023] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollar, P., and Girshick, R. (2023). Segment anything. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026. 33
- [Klingensmith et al., 2015] Klingensmith, M., Dryanovski, I., Srinivasa, S., and Xiao, J. (2015). CHISEL: Real time large scale 3D reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics: Science and Systems (RSS)*, Rome, Italy. 27
- [Kompis et al., 2021] Kompis, Y., Bartolomei, L., Mascaro, R., Teixeira, L., and Chli, M. (2021). Informed sampling exploration path planner for 3D reconstruction of large scenes. *IEEE Robotics and Automation Letters*, 6(4):7893–7900. 46
- [LaValle, 1998] LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Research report 9811, Department of Computer Science, Iowa State University. 35, 62, 65
- [Lee et al., 2010] Lee, T., Leok, M., and McClamroch, N. H. (2010). Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425. IEEE. 36
- [Leutenegger, 2022] Leutenegger, S. (2022). OKVIS2: Realtime scalable visual-inertial SLAM with loop closure. *arXiv preprint arXiv:2202.09199*. 38, 39, 58, 59, 101, 105
- [Leutenegger et al., 2015] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334. 38
- [Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440. 31

- [Lu et al., 2024] Lu, L., Zhang, Y., Zhou, P., Qi, J., Pan, Y., Fu, C., and Pan, J. (2024). Semantics-aware receding horizon planner for object-centric active mapping. *IEEE Robotics and Automation Letters*. 48
- [Luo et al., 2024] Luo, Y., Zhuang, Z., Pan, N., Feng, C., Shen, S., Gao, F., Cheng, H., and Zhou, B. (2024). Star-Searcher: A complete and efficient aerial system for autonomous target search in complex unknown environments. *IEEE Robotics and Automation Letters*. 48
- [Maggio et al., 2024] Maggio, D., Chang, Y., Hughes, N., Trang, M., Griffith, D., Dougherty, C., Cristofalo, E., Schmid, L., and Carlone, L. (2024). Clio: Real-time task-driven open-set 3D scene graphs. *IEEE Robotics and Automation Letters*, 9(10):8921–8928. 34
- [Markley et al., 2007] Markley, F. L., Cheng, Y., Crassidis, J. L., and Oshman, Y. (2007). Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197. 62
- [Matsuki et al., 2024] Matsuki, H., Murai, R., Kelly, P. H., and Davison, A. J. (2024). Gaussian splatting SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18039–18048. 30
- [McCormac et al., 2018] McCormac, J., Clark, R., Bloesch, M., Davison, A., and Leutenegger, S. (2018). Fusion++: Volumetric object-level SLAM. In *International Conference on 3D Vision (3DV)*, pages 32–41, Verona, Italy. 32, 84, 85
- [McCormac et al., 2017] McCormac, J., Handa, A., Davison, A. J., and Leutenegger, S. (2017). SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, Singapore. IEEE. 31, 84
- [Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421. 29
- [Millane et al., 2018] Millane, A., Taylor, Z., Oleynikova, H., Nieto, J., Siegwart, R., and Cadena, C. (2018). C-blox: A scalable and consistent TSDF-based dense mapping approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 995–1002. IEEE. 27, 28, 103
- [Mourikis and Roumeliotis, 2007] Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572. IEEE. 38

- [Müller et al., 2022] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15. 30, 31
- [Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163. 39
- [Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262. 39
- [Nash et al., 2010] Nash, A., Koenig, S., and Tovey, C. (2010). Lazy Theta*: Any-angle path planning and path length analysis in 3D. In *AAAI Conference on Artificial Intelligence*, pages 147–154. 42
- [Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE. 26, 27, 56
- [Nießner et al., 2013] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11. 27
- [Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528. 31
- [Oleynikova et al., 2016a] Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., and Galceran, E. (2016a). Continuous-time trajectory optimization for online UAV replanning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339. IEEE. 35
- [Oleynikova et al., 2016b] Oleynikova, H., Millane, A., Taylor, Z., Galceran, E., Nieto, J., and Siegwart, R. (2016b). Signed distance fields: A natural representation for both mapping and planning. In *Robotics: Science and Systems (RSS) Workshop: Geometry and beyond - representations, physics, and scene understanding for robotics*. 28
- [Oleynikova et al., 2017] Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., and Nieto, J. (2017). Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 27, 28, 32, 77

- [Oleynikova et al., 2018a] Oleynikova, H., Taylor, Z., Siegwart, R., and Nieto, J. (2018a). Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles. *IEEE Robotics and Automation Letters*, 3(3):1474–1481. 73, 92
- [Oleynikova et al., 2018b] Oleynikova, H., Taylor, Z., Siegwart, R., and Nieto, J. (2018b). Sparse 3D topological graphs for micro-aerial vehicle planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE. 47
- [Open Robotics, 2018a] Open Robotics (2018a). Gazebo Classic. <https://classic.gazebosim.org>. 78
- [Open Robotics, 2018b] Open Robotics (2018b). ROS Melodic Morenia. <https://wiki.ros.org/melodic>. 75
- [Open Robotics, 2020] Open Robotics (2020). ROS Noetic Ninjemys. <https://wiki.ros.org/noetic>. 94
- [Open Robotics, 2021] Open Robotics (2021). Gazebo Fortress. <https://gazebosim.org>. 109
- [Palazzolo and Stachniss, 2018] Palazzolo, E. and Stachniss, C. (2018). Effective exploration for MAVs based on the expected information gain. *Drones*, 2(1):9. 43, 67
- [Pan et al., 2022] Pan, Y., Kompis, Y., Bartolomei, L., Mascaro, R., Stachniss, C., and Chli, M. (2022). Voxfield: Non-projective signed distance fields for online planning and 3D reconstruction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5331–5338. 28
- [Pantic et al., 2022] Pantic, M., Cadena, C., Siegwart, R., and Ott, L. (2022). Sampling-free obstacle gradients and reactive planning in neural radiance fields (NeRF). In *IEEE International Conference on Robotics and Automation (ICRA): Workshop on Motion Planning with Implicit Neural Representations of Geometry*. 30
- [Papachristos et al., 2017] Papachristos, C., Khattak, S., and Alexis, K. (2017). Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE. 44, 67
- [Papachristos et al., 2019] Papachristos, C., Khattak, S., Mascarich, F., and Alexis, K. (2019). Autonomous navigation and mapping in underground mines using aerial robots. In *IEEE Aerospace Conference*, pages 1–8. IEEE. 44

- [Papatheodorou et al., 2025] Papatheodorou, S., Boche, S., Barbas Laina, S., and Leutenegger, S. (2025). Efficient submap-based autonomous MAV exploration using visual-inertial SLAM configurable for LiDARs or depth cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, Atlanta, USA. 20, 101
- [Papatheodorou et al., 2023] Papatheodorou, S., Funk, N., Tzoumanikas, D., Choi, C., Xu, B., and Leutenegger, S. (2023). Finding things in the unknown: Semantic object-centric exploration with an MAV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3345, London, United Kingdom. 18, 37, 49, 83
- [PX4, 2012] PX4 (2012). PX4 Autopilot, PX4 development team and community. <https://px4.io>. 109
- [Qin et al., 2019] Qin, T., Cao, S., Pan, J., and Shen, S. (2019). A general optimization-based framework for global pose estimation with multiple sensors. *arXiv preprint arXiv:1901.03642*. 39
- [Radford et al., 2021] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Aspell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *38th International Conference on Machine Learning (ICML)*, pages 8748–8763. 33
- [Ramezani et al., 2020] Ramezani, M., Wang, Y., Camurri, M., Wisth, D., Mattamala, M., and Fallon, M. (2020). The newer college dataset: Handheld LiDAR, inertial and vision with ground truth. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4353–4360. 25
- [Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. 83
- [Reijgwart et al., 2019] Reijgwart, V., Millane, A., Oleynikova, H., Siegwart, R., Cadena, C., and Nieto, J. (2019). Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters*, 5(1):227–234. 27, 28, 47, 101, 103, 110
- [Ren et al., 2022] Ren, Y., Zhu, F., Liu, W., Wang, Z., Lin, Y., Gao, F., and Zhang, F. (2022). Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6332–6339. IEEE. 36
- [Rosinol et al., 2020a] Rosinol, A., Abate, M., Chang, Y., and Carlone, L. (2020a). Kimera: An open-source library for real-time metric-semantic localization and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 31, 84

- [Rosinol et al., 2020b] Rosinol, A., Gupta, A., Abate, M., Shi, J., and Carlone, L. (2020b). 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems (RSS)*, Corvalis, Oregon, USA. 33
- [Rouček et al., 2019] Rouček, T., Pecka, M., Čížek, P., Petříček, T., Bayer, J., Šalanský, V., Heřt, D., Petrлік, M., Báča, T., Spurný, V., et al. (2019). DARPA subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systems (MESAS)*, pages 274–290, Palermo, Italy. 83
- [Salas-Moreno et al., 2013] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. J., and Davison, A. J. (2013). SLAM++: Simultaneous localisation and mapping at the level of objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359, Portland, OR, USA. IEEE. 84
- [Savva et al., 2019] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. (2019). Habitat: A platform for embodied AI research. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9339–9347, Seoul, Korea. IEEE. 84, 94
- [Schmid et al., 2024] Schmid, L., Abate, M., Chang, Y., and Carlone, L. (2024). Khronos: A unified approach for spatio-temporal metric-semantic SLAM in dynamic environments. In *Robotics: Science and Systems*, Delft, Netherlands. 119
- [Schmid et al., 2022] Schmid, L., Cheema, M. N., Reijgwart, V., Siegwart, R., Tombari, F., and Cadena, C. (2022). SC-Explorer: Incremental 3D scene completion for safe and efficient exploration mapping and planning. *arXiv preprint arXiv:2208.08307*. 49
- [Schmid et al., 2020] Schmid, L., Pantic, M., Khanna, R., Ott, L., Siegwart, R., and Nieto, J. (2020). An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507. 44, 47, 49
- [Schmid et al., 2021] Schmid, L., Reijgwart, V., Ott, L., Nieto, J., Siegwart, R., and Cadena, C. (2021). A unified approach for autonomous volumetric exploration of large scale environments under severe odometry drift. *IEEE Robotics and Automation Letters*, 6(3):4504–4511. 47, 102, 103, 110
- [Schneider et al., 2018] Schneider, T., Dymczyk, M., Fehr, M., Egger, K., Lynen, S., Gilitschenski, I., and Siegwart, R. (2018). Maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425. 38

- [Scott and Jerath, 2018] Scott, M. and Jerath, K. (2018). Multi-robot exploration and coverage: Entropy-based adaptive maps with adjacency control laws. *Annual American Control Conference (ACC)*, pages 4403–4408. 67
- [Selin et al., 2019] Selin, M., Tiger, M., Duberg, D., Heintz, F., and Jensfelt, P. (2019). Efficient autonomous exploration planning of large-scale 3-D environments. *IEEE Robotics and Automation Letters*, 4(2):1699–1706. 46, 67, 72, 76, 78, 80, 93, 103
- [Senarathne and Wang, 2016] Senarathne, P. and Wang, D. (2016). Towards autonomous 3D exploration using surface frontiers. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 34–41. IEEE. 46
- [Shan et al., 2021] Shan, T., Englot, B., Ratti, C., and Rus, D. (2021). LVI-SAM: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5692–5698. IEEE. 39, 101
- [Shen et al., 2012] Shen, S., Michael, N., and Kumar, V. (2012). Autonomous indoor 3D exploration with a micro-aerial vehicle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–15. IEEE. 41
- [Stachniss et al., 2005] Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using Rao-Blackwellized particle filters. In *Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA. 45
- [Steinbrücker et al., 2014] Steinbrücker, F., Sturm, J., and Cremers, D. (2014). Volumetric 3D mapping in real-time on a CPU. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2021–2028. IEEE. 27
- [Stepanas et al., 2022] Stepanas, K., Williams, J., Hernández, E., Ruetz, F., and Hines, T. (2022). OHM: GPU based occupancy map generation. *IEEE Robotics and Automation Letters*, 7(4):11078–11085. 26
- [Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE. 143
- [Sucan et al., 2012] Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82. 63, 72, 91
- [Tabib et al., 2016] Tabib, W., Corah, M., Michael, N., and Whittaker, R. (2016). Computationally efficient information-theoretic exploration of pits and caves. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3722–3727. IEEE. 43

- [Tordesillas and How, 2022] Tordesillas, J. and How, J. P. (2022). MINVO basis: Finding simplexes with minimum volume enclosing polynomial curves. *Computer-Aided Design*, 151. 34
- [Tranzatto et al., 2022] Tranzatto, M., Miki, T., Dharmadhikari, M., Bernreiter, L., Kulkarni, M., Mascarich, F., Andersson, O., Khattak, S., Hutter, M., Siegwart, R., et al. (2022). Cerberus in the DARPA subterranean challenge. *Science Robotics*, 7(66). 83
- [Tzoumanikas et al., 2019] Tzoumanikas, D., Li, W., Grimm, M., Zhang, K., Kovac, M., and Leutenegger, S. (2019). Fully autonomous micro air vehicle flight and landing on a moving target using visual-inertial estimation and model-predictive control. *Journal of Field Robotics*, 36(1):49–77. 37, 107
- [Tzoumanikas et al., 2020a] Tzoumanikas, D., Yan, Q., and Leutenegger, S. (2020a). Nonlinear MPC with motor failure identification and recovery for safe and aggressive multicopter flight. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8538–8544. IEEE. 36
- [Tzoumanikas et al., 2020b] Tzoumanikas, D., Yan, Q., and Leutenegger, S. (2020b). Nonlinear MPC with motor failure identification and recovery for safe and aggressive multicopter flight. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8538–8544, Paris, France. IEEE. 94
- [Usenko et al., 2019] Usenko, V., Demmel, N., Schubert, D., Stücker, J., and Cremers, D. (2019). Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429. 101
- [Usenko et al., 2017] Usenko, V., Von Stumberg, L., Pangercic, A., and Cremers, D. (2017). Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 215–222. IEEE. 34
- [Vasquez-Gomez et al., 2013] Vasquez-Gomez, J. I., Sucar, L. E., and Murrieta-Cid, R. (2013). Hierarchical ray tracing for fast volumetric next-best-view planning. *International Conference on Computer and Robot Vision*, pages 181–187. 67
- [Vespa et al., 2019] Vespa, E., Funk, N., Kelly, P. H., and Leutenegger, S. (2019). Adaptive-resolution octree-based volumetric SLAM. In *International Conference on 3D Vision (3DV)*, pages 654–662, Québec, Canada. 55, 56, 65, 89
- [Vespa et al., 2018] Vespa, E., Nikolov, N., Grimm, M., Nardi, L., Kelly, P. H., and Leutenegger, S. (2018). Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151. 26, 27, 53, 54, 55, 56, 57, 69, 70

- [Vizzo et al., 2023] Vizzo, I., Guadagnino, T., Mersch, B., Wiesmann, L., Behley, J., and Stachniss, C. (2023). KISS-ICP: In defense of point-to-point ICP—simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 8(2):1029–1036. 39
- [Wang et al., 2021a] Wang, Y., Funk, N., Ramezani, M., Papatheodorou, S., Popović, M., Camurri, M., Leutenegger, S., and Fallon, M. (2021a). Elastic and efficient LiDAR reconstruction for large-scale exploration tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5041, Xi’an, China. 101
- [Wang et al., 2021b] Wang, Y., Funk, N., Ramezani, M., Papatheodorou, S., Popović, M., Camurri, M., Leutenegger, S., and Fallon, M. (2021b). Elastic and efficient LiDAR reconstruction for large-scale exploration tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5041. IEEE. 60
- [Wang et al., 2022] Wang, Y., Ramezani, M., Mattamala, M., Digumarti, S. T., and Fallon, M. (2022). Strategies for large scale elastic and semantic LiDAR reconstruction. *Robotics and Autonomous Systems*, 155:104185. 103
- [Webb and Van Den Berg, 2013] Webb, D. J. and Van Den Berg, J. (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5054–5061. IEEE. 35
- [Westover, 1991] Westover, L. A. (1991). *Splatting: A parallel, feed-forward volume rendering algorithm*. PhD thesis, The University of North Carolina at Chapel Hill, Chapel Hill, NC, United States. 30
- [Whelan et al., 2012] Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J. J., and McDonald, J. (2012). Kintinuous: Spatially extended KinectFusion. In *Robotics: Science and Systems (RSS) Workshop on RGB-D: Advanced Reasoning with Depth Cameras*. 26
- [Whelan et al., 2015] Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., and Davison, A. J. (2015). ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems (RSS)*, Rome, Italy. 31
- [Williams et al., 2020] Williams, J., Jiang, S., O’Brien, M., Wagner, G., Hernandez, E., Cox, M., Pitt, A., Arkin, R., and Hudson, N. (2020). Online 3D frontier-based UGV and UAV exploration using direct point cloud visibility. In *IEEE International conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 263–270. 83

- [Witting et al., 2018] Witting, C., Fehr, M., Bähnemann, R., Oleynikova, H., and Siegwart, R. (2018). History-aware autonomous exploration in confined environments using MAVs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. 44, 67
- [Wu et al., 2021] Wu, S.-C., Wald, J., Tateno, K., Navab, N., and Tombari, F. (2021). SceneGraphFusion: Incremental 3D scene graph prediction from RGB-D sequences. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7515–7525. 33
- [Xu et al., 2022] Xu, B., Davison, A. J., and Leutenegger, S. (2022). Learning to complete object shapes for object-level mapping in dynamic scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2257–2264. 32
- [Xu et al., 2019] Xu, B., Li, W., Tzoumanikas, D., Bloesch, M., Davison, A., and Leutenegger, S. (2019). MID-Fusion: Octree-based object-level multi-instance dynamic SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 32, 85, 89, 90, 119
- [Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, Monterey, CA, USA. IEEE. 40, 45, 66, 83, 86, 101
- [Yu et al., 2021] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A. (2021). Plenotrees for real-time rendering of neural radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5752–5761. 29
- [Yu et al., 2023] Yu, J., Shen, H., Xu, J., and Zhang, T. (2023). ECHO: An efficient heuristic viewpoint determination method on frontier-based autonomous exploration for quadrotors. *IEEE Robotics and Automation Letters*, 8(8):5047–5054. 46
- [Zeng et al., 2023] Zeng, J., Li, Y., Ran, Y., Li, S., Gao, F., Li, L., He, S., Chen, J., and Ye, Q. (2023). Efficient view path planning for autonomous implicit reconstruction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4063–4069, London, United Kingdom. IEEE. 43
- [Zeng et al., 2024] Zeng, J., Li, Y., Sun, J., Ye, Q., Ran, Y., and Chen, J. (2024). Autonomous implicit indoor scene reconstruction with frontier exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 18041–18047, Yokohama, Japan. IEEE. 47
- [Zeng et al., 2013] Zeng, M., Zhao, F., Zheng, J., and Liu, X. (2013). Octree-based fusion for realtime 3D reconstruction. *Graphical Models*, 75(3):126–136. 27

- [Zeng et al., 2020] Zeng, Z., Röfer, A., and Jenkins, O. C. (2020). Semantic linking maps for active visual object search. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1984–1990, Paris, France. IEEE. 48, 83, 85
- [Zhang and Singh, 2014] Zhang, J. and Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems (RSS)*, Berkeley, CA, USA. 39
- [Zhang et al., 2024] Zhang, Y., Chen, X., Feng, C., Zhou, B., and Shen, S. (2024). FALCON: Fast autonomous aerial exploration using coverage path guidance. *IEEE Transactions on Robotics*, 41:1365–1385. 42
- [Zhang et al., 2022] Zhang, Y., Zhou, B., Wang, L., and Shen, S. (2022). Exploration with global consistency using real-time re-integration and active loop closure. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9682–9688. IEEE. 103
- [Zhao et al., 2023] Zhao, X., Ding, W., An, Y., Du, Y., Yu, T., Li, M., Tang, M., and Wang, J. (2023). Fast segment anything. *arXiv preprint arXiv:2306.12156*. 34
- [Zheng et al., 2022] Zheng, C., Zhu, Q., Xu, W., Liu, X., Guo, Q., and Zhang, F. (2022). FAST-LIVO: Fast and tightly-coupled sparse-direct lidar-inertial-visual odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4003–4009. IEEE. 39, 101
- [Zhong et al., 2022] Zhong, P., Chen, B., Lu, S., Meng, X., and Liang, Y. (2022). Information-driven fast marching autonomous exploration with aerial robots. *IEEE Robotics and Automation Letters*, 7(2):810–817. 46
- [Zhou et al., 2021] Zhou, B., Zhang, Y., Chen, X., and Shen, S. (2021). FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786. 42
- [Zhou et al., 2024] Zhou, S., Papatheodorou, S., Leutenegger, S., and Schoellig, A. P. (2024). Control-barrier-aided teleoperation with visual-inertial SLAM for safe MAV navigation in complex environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 17836–17842, Yokohama, Japan. 119
- [Zhou et al., 2020] Zhou, X., Wang, Z., Ye, H., Xu, C., and Gao, F. (2020). EGO-Planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485. 35
- [Zhu et al., 2015] Zhu, C., Ding, R., Lin, M., and Wu, Y. (2015). A 3D frontier-based exploration tool for MAVs. In *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 348–352. IEEE. 41

- [Zhu et al., 2021] Zhu, D., Wang, C., Wang, W., Garg, R., Scherer, S., and Meng, M. Q.-H. (2021). VDB-EDT: An efficient Euclidean distance transform algorithm based on VDB data structure. *arXiv preprint arXiv:2105.04419*. 28
- [Zucker et al., 2013] Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9–10):1164–1193. 35

Appendix

A.1 Happy Accidents

The images in this appendix were generated as a side-effect of the work described in this thesis.

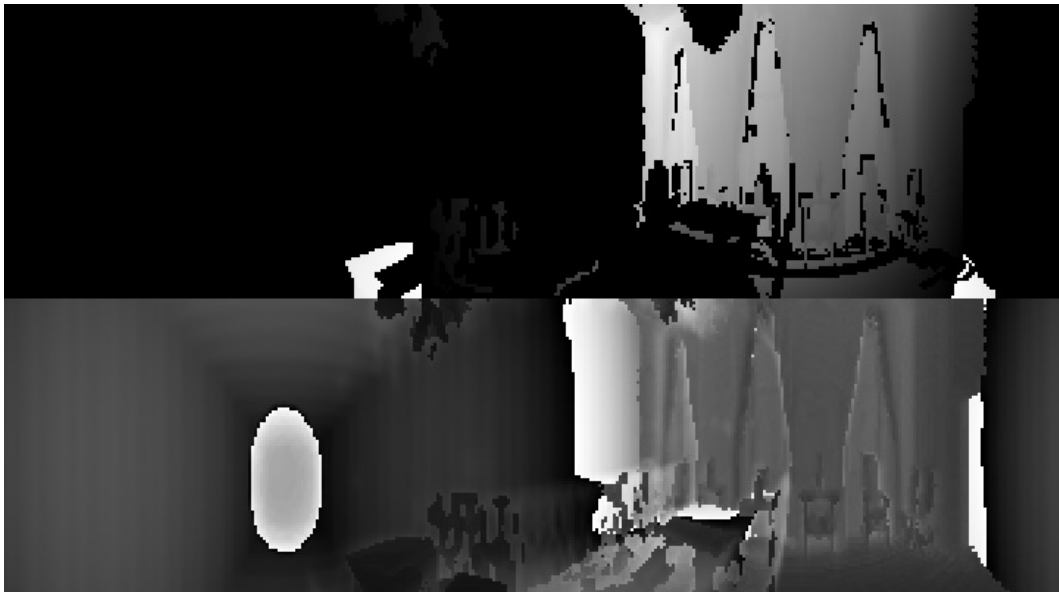


Figure A.1: 2spooky4me (2021): 360° depth (top half) and map entropy (bottom half) renders from an occupancy map of the King’s Hall in Skokloster Castle. The map was generated using images rendered from Habitat-Sim. Both renders are wrong.



Figure A.2: Lollipop (2021): Colour render from the 2t7WUuJeko7 scene of the Matterport3D dataset [Chang et al., 2017] overlaid with the corresponding depth render coloured with a rainbow colourmap. Used to check whether the colour and depth renders were aligned. Turns out they were aligned and this was not the cause of whatever bug was experienced at the time.



Figure A.3: Modern Architecture (2022): Visualisation of the ground-truth colour and instance ID meshes from the 2t7WUuJeko7 scene of the Matterport3D dataset [Chang et al., 2017]. A prime example of the most common issue in robotics, a transformation being wrong.

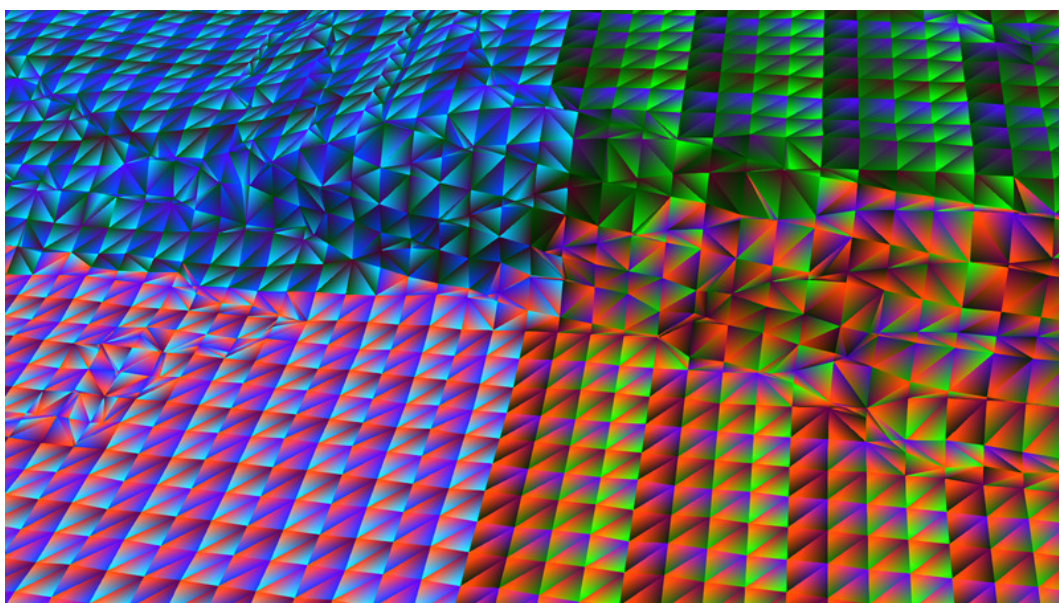


Figure A.4: Uninitialised (2022): 3D mesh reconstruction of the `fr1/desk` sequence from the TUM RGB-D dataset [Sturm et al., 2012] using uninitialised memory for the vertex colours. Using uninitialised memory leads to surprising results indeed.