Imperial College London

Department of Computing

Multi-resolution Mapping and Planning for UAV Navigation in Attitude Constrained Environments

Nils Funk

9th October 2023

Supervised by Dr Stefan Leutenegger

Second supervisor: Professor Aldo Faisal

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.

Abstract

In this thesis we aim to bridge the gap between high quality map reconstruction and Unmanned Aerial Vehicles (UAVs) SE(3) motion planning in challenging environments with narrow openings, such as disaster areas, which requires attitude to be considered. We propose an efficient system that leverages the concept of adaptive-resolution volumetric mapping, which naturally integrates with the hierarchical decomposition of space in an octree data structure. Instead of a Truncated Signed Distance Function (TSDF), we adopt mapping of occupancy probabilities in log-odds representation, which allows representation of both surfaces, as well as the entire free, i.e. observed space, as opposed to unobserved space. We introduce a method for choosing resolution -on the fly- in real-time by means of a multi-scale max-min pooling of the input depth image. The notion of explicit free space mapping paired with the spatial hierarchy in the data structure, as well as map resolution, allows for collision queries, as needed for robot motion planning, at unprecedented speed. Our mapping strategy supports pinhole cameras as well as spherical sensor models. Additionally, we introduce a first-of-a-kind global minimum cost path search method based on A* that considers attitude along the path. State-of-the-art methods incorporate attitude only in the refinement stage. To make the problem tractable, our method exploits an adaptive and coarse-to-fine approach using global and local A* runs, plus an efficient method to introduce the UAV attitude in the process. We integrate our method with an SE(3) trajectory optimisation method based on a safe-flight-corridor, yielding a complete path planning pipeline. We quantitatively evaluate our mapping strategy in terms of mapping accuracy, memory, runtime performance, and planning performance showing improvements over the state-of-the-art, particularly in cases requiring high resolution maps. Furthermore, extensive evaluation is undertaken using the AirSim flight simulator under closed loop

control in a set of randomised maps, allowing us to quantitatively assess our path initialisation method. We show that it achieves significantly higher success rates than the baselines, at a reduced computational burden.

Acknowledgements

At this point I would like to express my gratitude to those who supported me through the course of my PhD.

My special gratitude goes to Dr Stefan Leutenegger for supervising me throughout my PhD. Your remarks and comments have been an enormous help. Thank you Stefan for making this all possible and giving me the opportunity to undertake my research under your supervision.

I am also really grateful for the opportunity Slamcore has provided me. I feel fortunate to have worked alongside so many talented researchers. A special mention goes to Pablo for co-supervising me throughout the years.

I would also like to point out the enormous support Juan has provided me. You cannot imagine how important your support has been to me. Your willingness to help, guide and discuss ideas with me at all times is what made this all possible. I will be forever grateful for this.

Thank you Sotiris for the amazing time and laughter we were able to share together. You have been a great friend in and outside of the lab.

Thank you Sanjay for proofreading literally every document I wrote during the last 10 years. I promise this was the last one.

Furthermore I would like to thank the entire Imperial Smart Robotics Lab. Thank you Binbin, Masha, Dimos and Chris for all the great moments we spent together in the lab.

I would also like to thank Iosina and Amani for the various support I received related to my PhD studies.

Lastly, I would like to thank my parents for supporting me at every step throughout my academic and personal life. I could not have asked for more. You are the best.

Nils Funk

London, UK, March, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents of Chapter

1.1 Motivation

The past decade has brought impressive advancements in the field of mobile robotics, fueled by the advent of RGB-D cameras and ever more powerful processors, including GPUs. Recent technological advances are inciting the use of mobile robots for exploration and monitoring tasks. Their growing versatility and autonomy offer safe solutions in a wide range of applications, including aerial surveillance, infrastructure inspection, and search and rescue [Nex and Remondino, 2014]. In real-world applications requiring vehicles to navigate unstructured, complex environments, finding suitable but often not obvious paths can mean the difference between success and failure of the mission.

Overall motion planning maintains a tight symbiosis with the mapping structure for collision avoidance. Unmanned aerial vehicles (UAVs) are especially suited to navigate in those regions due to their maneuverability and low cost. However, to fully exploit their potential, a key task is enabling light-weight platforms to operate autonomously with limited on-board computational resources, demanding the creation of algorithms specifically tailored to the mapping and motion planning requirements of the system. Therefore, an efficient mapping structure is required for fast high resolution mapping on computationally constrained platforms. There are several efficient mapping methods for UAV trajectory planning that rely on 3D volumetric representations. The main drawback of previous approaches is that their computational performance degrades drastically with the discretisation of the environment, since they operate on map data in the same way regardless of the occupancy status and geometry of the underlying space. As a result, these methods are only suitable for scenarios requiring coarse reconstructions or navigating in areas with relatively large obstacles. Therefore, our first aim is the creation of an efficient high-resolution mapping framework for robot motion planning.

However, even with an efficient map representation, fully autonomous UAV

motion planning is non-trivial, especially in highly constrained environments that require the consideration of the UAVs attitude along the trajectory. While motion planning in challenging environments has gained significant interest in recent years, autonomous navigation is still an unsolved research topic. The majority of work does not consider the UAVs attitude and therefore relies on a conservative sphere shape approximation of the robot, preventing the UAV from flying through narrow passages. Only recently has the first approach [Wang et al., 2022] for orientation aware SE(3) trajectory optimisation been published. However, significant limitations still remain: a list of carefully handpicked waypoints is required to initialise the framework and therefore restricts the pipeline to only partial autonomy. We argue that an efficient path search method in the space of position and orientation is the key to unlock the true power UAV of SE(3) path planning, enabling fast planning in unstructured environments.

In summary, this thesis addresses the challenge of trading off mapping accuracy against computational efficiency for planning in online, on-board robotic applications that require *both*. Furthermore, we focus on solving the missing parts for fully autonomous SE(3) UAV navigation, by providing an attitude aware path initialisation method that seamlessly and efficiently integrates with the mapping framework.

1.2 Thesis Structure and Contributions

The remainder of this thesis is structured as follows:

Chapter 2 provides an extensive overview of the state-of-the-art approaches for UAV environmental mapping and trajectory planning.

Chapter 3 introduces the basic notation and provides the technical background required. We present the pinhole and LiDAR camera models used and describe the

supereight data structure. We elaborate on A* graph-search and ESDF wave-front propagation algorithm. Lastly, an explanation of the Safe Flight Corridor generation and trajectory optimisation is provided.

Chapter 4 (Contribution I) presents our efficient *dense* volumetric multi-resolution mapping framework. The contribution comprises a data structure, fusion method and sensor models that seamlessly adapts the map to different scene scales. A novel fast map-to-image allocation algorithm and integration method together enable real-time online probabilistic occupancy mapping and accurate surface reconstruction. The adaptive map resolution is chosen according to sensor information, constraining the induced sampling error. Additionally, we introduce a new scale selection and data propagation scheme to keep the hierarchy consistent between levels [Vespa et al., 2019]. We explicitly differentiate between observed, occupied and *free* space, allowing multi-resolution occupancy queries. Our framework is adaptive to numerous sensor modalities and currently supports pinhole cameras as well as LiDAR sensor models. A comprehensive evaluation with respect to the state-of-the-art reveals vast improvements in the trade-offs of mapping accuracy, speed, memory consumption, tracking accuracy, and planning performance.

Chapter 5 (Contribution II) describes our path initialisation method target for global minimum cost path computation in orientation constrained maps. We present a hierarchical attitude-aware A* algorithm. An underlying adaptive resolution octree represents the target cost map and can be seamlessly generated from our 3D occupancy mapping pipeline. The adaptive-resolution of the search graph reduces the map discretisation significantly in places where it is not required. Furthermore, a novel coarse to fine strategy ensures the global search is done at a coarse resolution, only going to finer resolutions locally where needed. Additionally, in local areas where orientation is critical, we introduce an efficient method to consider the UAV's attitude during the search, based on pre-computed binary masks. Our planner is integration into a working pipeline for SE(3) UAV trajectory planning; hereby

achieving significantly faster computation times and higher success rates compared to the state-of-the-art. An extensive evaluation has been conducted in randomised environments using a drone racing simulator with a physics engine and closed loop control.

Chapter 6 summarises the overall result of the thesis and gives an outlook for future work.

1.3 Publications

The work described in this thesis resulted in the following publications:

- Nils Funk, Juan Tarrio, Sotiris Papatheodorou, Pablo F. Alcantarilla and Stefan Leutenegger. **Orientation Aware Hierarchical, Adaptive-Resolution A* Algorithm for SE(3) UAV Trajectory Planning**. *Currently under review for IEEE Robotics and Automation Letters (RA-L)*, 2023.

- Nils Funk, Juan Tarrio, Sotiris Papatheodorou, Marija Popović, Pablo F. Alcantarilla and Stefan Leutenegger. **Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning**. *IEEE Robotics and Automation Letters (RA-L)*. Vol. 6, No. 2, pp. 3553-3560, 2021. [Funk et al., 2021]

- Yiduo Wang, Nils Funk, Milad Ramezani, Sotiris Papatheodorou, Marija Popović, Marco Camurri, Stefan Leutenegger and Maurice Fallon. **Elastic and Efficient LiDAR Reconstruction for Large-Scale Exploration Tasks**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. [Wang et al., 2021a]

- Emanuele Vespa, Nils Funk, Paul H. J. Kelly and Stefan Leutenegger (2019). **Adaptive-Resolution Octree-Based Volumetric SLAM**. *International Conference on 3D Vision (3DV)*. [Vespa et al., 2019]

1.4 Funding

We would like to thank Slamcore for funding this work.

Chapter 2

Literature Review

The following chapter provides an overview of the related work. We divide the literature review into two sections for mapping for path planning and UAV motion planning.

Contents of Chapter

2.1 Mapping for 3D Motion Planning

A large body of literature addresses the problem of obtaining a dense 3D representation of the world. In this section, we overview recent work, focusing on volumetric reconstruction methods suitable for real-time robotic applications running on constrained hardware. This leaves aside most batch integration methods, where a map is only available for planning at the end of the mapping run.

A major milestone in online RGB-D 3D reconstruction systems is KinectFusion [Newcombe et al., 2011] (see Fig. 2.1), which enables dense volumetric modelling in real-time at sub-centimetre resolutions. However, it is limited to small, bounded environments, as the mapping is locked to a fixed volume with a pre-defined voxel resolution, and requires GPU processing to achieve real-time performance. To improve scalability, several extensions to the original algorithm have been proposed. One possibility is to use moving fixed-size sliding volumes [Whelan et al., 2012, Usenko et al., 2017] to achieve mapping in a dynamically growing space. Another strategy is to exploit memory-efficient data structures, such as octree-based voxel grids [Vespa et al., 2018, Zeng et al., 2013, Steinbrücker et al., 2014] or hash tables [Nießner et al., 2013, Klingensmith et al., 2015], for quicker spatial indexing.

Most recently VDBFusion [Vizzo et al., 2022] introduced a TSDF mapping framework based on the open-source OpenVDB library [Museth et al., 2013]. OpenVDB is commonly used in film making to render photo realistic scenes. The underlying VDB hierarchical data structure consists of a tree structure with numerous roots notes and leaf nodes comprising blocks of $8 \times 8 \times 8$ voxels, providing unbound space access and compact storage. VDBFuison [Vizzo et al., 2022] exploits the spare VDB data structure to incrementally integrate point clouds in a highly efficient manner, allowing the framework to work with numerous sensor modalities like RGB-D and LiDAR sensors. The efficient implementation of the framework reduces memory

footprint while improving map accuracy compared to state-of-the-art approaches and runs on a single CPU core.

Despite impressive progress, most 3D mapping research has targeted the application of surface reconstruction, which aim to produce a high-quality mesh/point-cloud of a scene. From a navigation perspective, the concept of observed free space is equally or more important than surface accuracy. Unfortunately, most of these methods model free space only near the surface boundaries and do not generally distinguish between free and unvisited areas in initially unknown environments. Similarly [Gao and Shen, 2016] uses a point cloud map representation stored in a *kd-tree* only keeping track of occupied space, approximating all remaining space as free. Our work provides an explicit distinction between the two as necessary for robotic planning, exploration, and collision avoidance.

Figure 2.1: Dense reconstruction generated with Kinect Fusion, showing the raw point cloud (left), surface normal map (middle) and rendered surface (right) (Figure taken from [Newcombe et al., 2011]).

2.1.1 Euclidean Signed Distance Field

In the context of navigation, Euclidean Signed Distance Field (ESDF)-based mapping methods are commonly used for motion planning tasks as they provide distance information for trajectory optimisation strategies. An ESDF encodes the Euclidean distance from the nearest obstacle and thus can significantly simplify collision

queries. A distance value of 0 represents the surface crossing, while a positive and negative distance indicates that the voxel is located in front and behind the closest surface. [Felzenszwalb and Huttenlocher, 2012] introduces an envelope algorithm with $\mathcal{O}(n)$ complexity, computing the distance transform using parabolic curves. Unfortunately, the algorithm is not suitable for incremental updates. Similarly [Chen et al., 2022] presents a GPU accelerated framework that can be updated incrementally. Significant work has been done on the incremental construction of ESDF maps for 3D motion planning, including *voxblox* [Oleynikova et al., 2017] and *FIESTA* [Han et al., 2019], which construct ESDF maps from Truncated Signed Distance Field (TSDF) maps and occupancy maps respectively. *voxblox* [Oleynikova et al., 2017] produces a quasi-Euclidean truncated ESDF, while FIESTA [Han et al., 2019] keeps track of the closest surface voxel during the wavefront propagation to generate a more accurate non-truncated ESDF. However, as these methods are designed for fast on-board collision checking, they rely on coarsely discretised environments with voxel grid resolutions on the order of $\sim 20\,\text{cm}$ magnitude. In contrast, in this thesis our mapping approach is also motivated by applications like close-up inspection [Stent et al., 2015], which require detailed scene reconstructions. Recently, Voxfield [Pan et al., 2022] introduces a framework based on the formulation of non-projective truncated signed distance fields (TSDFs), providing ESDF values with sub-voxel accuracy while remaining computationally efficient.

2.1.2 Occupancy Mapping

An alternative representation for planning are occupancy maps [Elfes, 1989, Thrun, 2003]. In 3D, OctoMap [Hornung et al., 2013] is a popular framework that uses hierarchical octrees to track occupancy probabilities as sensor data is received. Similar to the original work of [Elfes, 1989], it uses an inverse sensor model that efficiently approximates the posterior using an additive log-odds update equation, which resembles the TSDF update procedure [Newcombe et al., 2011].

However, while OctoMap works well with sparse LiDAR data, its performance degrades significantly as map resolution increases, as well as with noisier sensors. Interestingly, a very recent contribution shows improvements over OctoMap in terms of memory and run time by adaptively downsampling the pointcloud and integrating free space at lower resolutions [Duberg and Jensfelt, 2020], highlighting the importance of this topic. While this method uses a set of distance-based rules to decide the integration resolution (similar to [Vespa et al., 2018]), leading in the fastest setups to non-conservative assignments of free space, our work tackles this problem by rigorously assessing the probabilistic (inverse) measurement model, introducing an approach for choosing a sampling resolution that ensures these errors do not happen, while also improving run time performance.

Closest to our work, [Vespa et al., 2018] proposed an efficient pipeline with an octree-based implementation for reconstruction and planning. Their approach supports either TSDF-based or occupancy mapping using the spline inverse sensor model of [Loop et al., 2016]. However, the use of multi-resolution is very basic and multiple assumptions limit its applicability to planning.

Subsequent work [Vespa et al., 2019] extended this system to handle data integration with varying levels of detail and rendering at multiple resolution scales using TSDF maps, but limited to surface reconstruction. In the methods developed in this thesis, we have drawn inspiration from this approach in terms of data structure and propagation; however, our main focus has been on volumetric occupancy-based representations for planning and space understanding, where the goal is to probabilistically classify all observed space into occupied, free, or unknown at high resolutions, while also providing a high quality (surface) reconstruction of the environment.

A comparison of occupancy, TSDF and ESDF map representation is shown in Fig. 2.2.

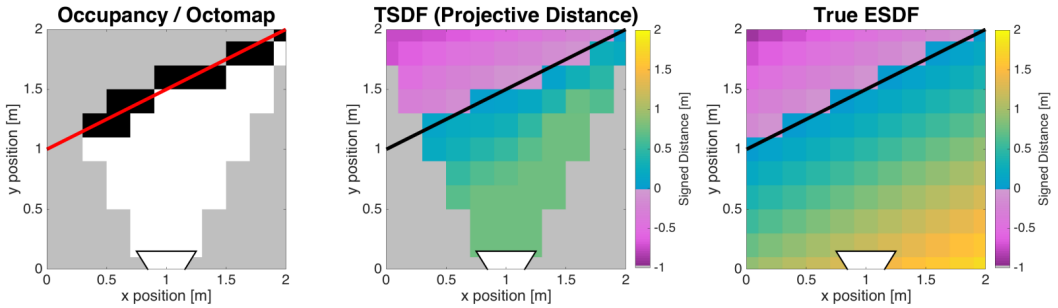

Figure 2.2: Illustration of different map representations. (a) Occupancy map: Occupied, free and unknown space are represented in black, white and grey respectively. (b) TSDF: Unobserved space is shown in grey. The projective distance follows along the ray and does not represent the actual distance to the surface. (c) ESDF: The distance represents the actual distance to the closest surface point (All figures taken from [Oleynikova et al., 2016b]).

2.1.3 Implicit Neural Reconstruction

Implicit neural reconstructions provide an implicit 3D scene representation, utilising latent features in a high-dimensional space using a neural network [Park et al., 2019, Mescheder et al., 2019]. In recent years special attention has been brought to Neural Radiance Fields (NeRF) [Mildenhall et al., 2020] as they improve map accuracy at a lower memory footprint and provide further novel view synthesis. Newer methods build up on NeRF for incremental dense mapping. However the networks require a long time for training and the implicit collision checking is much slower than the explicit volumetric approach, rendering them unsuitable for path planning. To approach these challenges, hybrid representations have been introduced. To accelerate the scene geometry initialisation and training time, [Jiang et al., 2023] introduces a hierarchical hybrid representation with an implicit multi-resolution hash encoding aided by explicit octree SDF priors at different levels of detail. Alternatively, [Zeng et al., 2023] combines the implicit fine-grained representation with coarse volumetric representations for fast collision checks and view-point filtering.

2.2 UAV Motion Planning

Numerous methods for UAV motion planning have been proposed over the years, tackling domain specific tasks like the exploration of unknown spaces [Schmid et al., 2021, Zhong et al., 2022], the traversal of dynamic environments [Wang et al., 2021b, Kong et al., 2021, Guo et al., 2022], trajectory execution safety [Tordesillas et al., 2022, Zhou et al., 2021a] and the computation of spacial- and time-optimal trajectories in challenging environments [Hanover et al., 2023]. While those domains show large overlap in their requirements to compute kinodynamically feasible and collision free trajectories, their individual research focus differs significantly. In the following we will further introduce the state-of-the-art approaches for the latter, traversing complex environments in an efficient and safe manner.

2.2.1 Deep Learning Based Trajectory Optimisation

Deep learning based approaches have shown success in drone racing challenges. Initially convolutional neural networks (CNNs) were used in combination with Model Predictive Control [Kaufmann et al., 2018] to navigate a quadrotor through a racing course with multiple gates. While the CNN predicts the 3D poses of the gate's center, the navigation strategy for the quadrotor is still based on the MPC outputs control commands.

More recent approaches use deep learning to infer control strategies, directly mapping sensory data to control outputs. [Song et al., 2021] uses reinforcement learning and relative gate observations to generate near-time-optimal trajectories. While the approach shows promising results in simulation, the application in real-world experiments report high tracking errors. Additionally none of these works focus on collision avoidance.

Using a classical topological guiding path along a sequence of waypoints [Penicka et al., 2022] introduces a deep reinforcement learning framework to combine the

computation of minimum-time flight trajectories and obstacle avoidance. To overcome the need of a reference path [Song et al., 2022] presents a perception-aware deep learning based planner, combining imitation learning with reinforcement learning for agile flight in cluttered environments. However the navigation strategy is purely reactive, taking the camera's current field-of-view into account.

2.2.2 Motion Primitives

Planners based on pre-computed motion primitives allow for locally highly optimised trajectories. [Lopez and How, 2017a, Lopez and How, 2017b] uses a relaxed-constraint Model Predictive Control (MPC) framework that chooses a local trajectory from a set of previously generated motion primitive. While the framework allows for the fast reaction to instantaneous perception data, it only considers a local planning horizon using a *kd*-tree representation of the local environment. [Zhang et al., 2019, Zhang et al., 2020] uses a hierarchical state lattice of pre-computed cubic spline motion primitives, poorly approximating the UAV dynamics. Obstacles inside the sensor range are considered deterministic while those outside are regarded probabilistic. Based on the prior map information and sensor data the framework chooses a motion primitive to the sensor frontier, maximising the likelihood of the UAV to reach the goal considering the collision probability of the entire path. Alternatively [Liu et al., 2017a] computes a global path using a search-based approach, concatenating a series of short-duration motion primitives. The primitives are pre-computed by solving an optimal control problem. However in all cases, the limited set of motion primitives makes it difficult to manoeuvre highly constrained environments, while an increase in primitives suffers from a combinatorial explosion.

In [Mellinger and Kumar, 2011], the safe trajectory generation is formulated as an optimisation problem exploiting the differential flatness of the quadrotor dynamics. Following a common quadrotor model, [Mellinger and Kumar, 2011]

shows that the UAV is able to follow any C^4 smooth continuous-time function, assuming the velocity and acceleration limitations are not violated. Choosing a time-parameterised polynomial for each dimension and segment between given waypoints provides a simple but powerful motion primitive. The work computes minimum-snap trajectories by iteratively optimising the polynomial coefficients and segment times, switching between spatial and temporal optimisation. To enforce continuity between polynomial-segments equality constraints of the end-point derivatives are added to the quadratic program. [Richter et al., 2016] formulates the unconstrained dual of the optimisation. By expressing the polynomials based on their end-point derivatives, the equality constraints can be integrated into the cost function. While neither of the works focus on collision avoidance, they build an important foundation for two branches of UAV trajectory optimisation in cluttered environments over the past decade.

A large portion of research focusing on UAV motion planning in complex environments differentiates between gradient-based trajectory optimisation and trajectory optimisation in a Safe Flight Corridor (SFC). Generally both approaches combine a two stage process of trajectory / path initialisation and iterative spatial and temporal trajectory optimisation. While those works are all based on polynomial motion primitives, different polynomial representation have been suggested. In [Gao et al., 2018] Bezier curves assembled from numerous Bernstein polynomials are used as a motion primitive. Bezier curves facilitate a dual to the standard polynomial representation by substituting the polynomial coefficients with control points. The advantage of Bezier curves is that they provide a convex hull property, meaning that the curve will never surpass the convex hull spanned by its control points. However the curve's ability to expand into the entire convex space is limited, causing the trajectories to be conservative. To reduce this problem, [Tordesillas and How, 2022] introduces MINVO basis, a polynomial representation with similar properties to Bernstein polynomials with a better minimum volume enclosing.

Alternatively, [Usenko et al., 2017] introduces the use of uniform B-splines for UAV motion planning. B-splines combine the convex hull property with the property of local control. This property states that the change of a control point only influences the spline locally. However a disadvantage of B-splines is that they do not pass through their control points, making motion planning through dedicated waypoints difficult. Furthermore, their volume enclosing is significantly reduced in contrast to Bernstein polynomials, limiting its optimisation potential in highly cluttered environments. A comparison of the Bernstein, MINVO and B-Spline polynomial representations and their convex hulls is shown in Fig. 2.3.

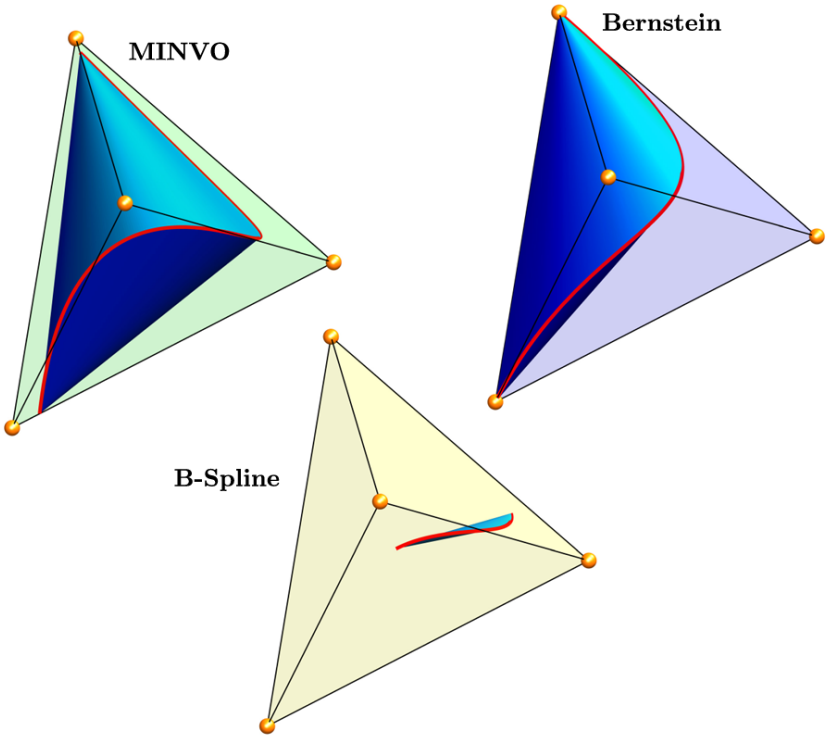

Figure 2.3: The MINVO, Bernstein and B-Spline representations showing a 3rd order polynomial for the same control points (All figures taken from [Tordesillas and How, 2022]).

2.2.3 Gradient-based Trajectory Optimisation

Gradient-based trajectory optimisation methods incorporate a collision cost into the objective function and use the cost gradient along the trajectory iteratively to minimise the overall cost. A large variety of gradient-based approaches have been published that mainly differ in the initialisation approach of the trajectory, the polynomial representation used, the way the collision cost gradient is computed and the chosen optimisation method.

Based on CHOMP's [Zucker et al., 2013] gradient-based trajectory optimisation for discrete-time trajectory, [Oleynikova et al., 2016a] introduces a continuous-time trajectory optimisation method using piece-wise polynomial motion primitives. The method uses a discretised ESDF map representation to compute the cost gradient based on a distance dependent potential function. To approximate the collision cost of the trajectory's line integral, the trajectory is sampled according to the discretisation of the map. Additionally the gradient in each sample point is computed and the trajectory iteratively optimised with a quasi-Newton method. However, the approach's success rate is heavily dependent on the initialisation of the polynomial parameters and improved by running random restarts. This is due to the fact that gradient optimisation is prone to get stuck, if the initial trajectory passes through an obstacle. Therefore a collision-free initialisation is needed in the majority of cases. In [Zhou et al., 2020] and [Zhou et al., 2021a] a uniform B-spline trajectory representation is used. To prevent the trajectory from getting stuck in obstacles during the gradient optimisation, the author guides the optimisation with a collision free straight line path. Similarly [Zhou et al., 2019] improves the efficiency of the gradient-based optimisation utilising the convex hull property of B-splines, after initialising the trajectory with a kinodynamically feasible sampled path. Alternatively, [Zhou et al., 2021b] and [Ding et al., 2019] provide gradient-base frameworks that to not require the computation of an ESDF.

Nevertheless, while gradient-based trajectory optimisation methods allow the exploitation of the entire free-space, they are prone to getting stuck in local minima.

2.2.4 Safe Flight Corridor Based Trajectory Optimisation

Contrary to gradient-based optimisation, frameworks based on a Safe Flight Corridor (SFC) use a conservative approximation of the available free-space, modelled as a collection of overlapping convex shapes. Generally the state-of-the-art approaches differ in the initialisation of the Safe Flight Corridor construction, the convex shape primitive chosen, the used polynomial representation and the constraints on the segment end points.

All methods compute an initial path from the start to goal position, from which the convex free-space approximation is generated. Given a set of way-points along the path, a convex shape is expanded around each path segment. Various shape primitives have been tested, including axis aligned cuboids [Chen et al., 2016], polyhedra [Deits and Tedrake, 2015, Liu et al., 2017b, Gao et al., 2020] and spheres [Gao and Shen, 2016, Ren et al., 2022] (see Fig. 2.4) adding 1D linear, 3D linear and 3D quadratic inequality constraints to the optimisation respectively. [Chen et al., 2016] uses a standard polynomial representation as a motion primitive and iteratively optimises the trajectory. In an initial step, no constraints are added to the optimisation. After each iteration the corridor violation are computed, which can easily be done by finding the extrema of the polynomial in each dimension. For each violation a inequality constraint is added to the optimisation. The step is repeated until all corridor violations are removed. In [Gao et al., 2018] a Bernstein polynomial representation is used and the safety of the trajectory guaranteed by constraining the control points to stay within the SFC. Similarly, [Gao et al., 2020] constraints the Bernstein polynomial control points using the hyperplane constraints of a SFC assemble form polyhedra.

While Safe Flight Corridor based approaches do not get stuck in local minima, the trajectory is only optimised within the corridor, neglecting the potential of the remaining free space.

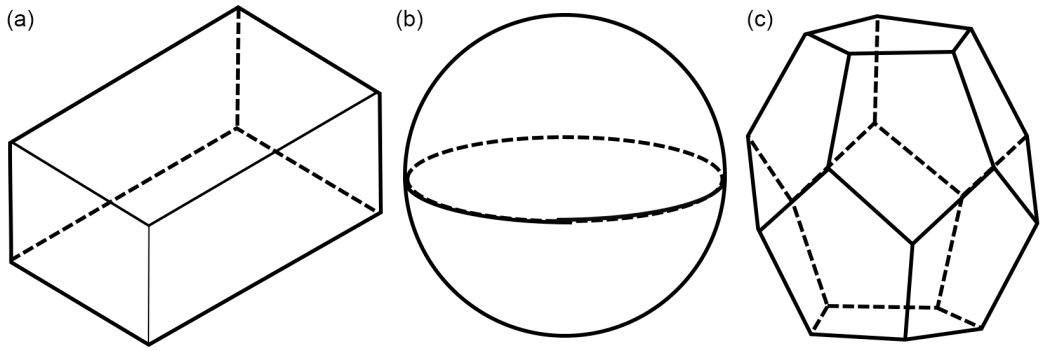

Figure 2.4: Safe Flight Corridor shape primitives: (a) Axis Aligned Cuboid, (b) Sphere, (c) Polyhedra

2.2.5 Path Initialisation

One fundamental part of gradient-based and Safe Flight Corridor based methods is the initialisation step. The overall success and quality of the optimised trajectory is heavily dependent on this stage. We give a combined overview of the state-of-the-art trajectory and path initialisation frameworks for both optimisation approach as they share a lot of similar approaches.

The majority of methods classify into sampling-based and search-based methods and do not (or only partially) consider the kinodynamic feasibility of the trajectory at this stage. They mainly aim to provide a collision free initial geometric path from the start to goal position, assuming a spherical UAV shape approximation.

Rapidly-exploring Random Trees (RRT) [LaValle, 1998] are a popular sampling-based approach and especially suited to search high-dimensional spaces. The tree is incrementally built by adding collision free path segments to randomly drawn sample points. The sampling strategy is generally biased to expand into large areas of free-space. In [Karaman and Frazzoli, 2010, Karaman and Frazzoli, 2011] RRT*,

an asymptotically optimal version of RRT is presented. Informed RRT* [Gammell et al., 2014] further improves the convergence rate of RRT* while maintaining its probabilistic guarantees on optimality and completeness, incorporating heuristics into the sampling strategy. The method is used in [Gao et al., 2017] to improve the sampling efficiency and quality of the trajectory initialisation. Rather than using straight line path segments to sample points, Kinodynamic RRT* [Webb and van den Berg, 2013] connects any two states with controllable linear dynamics with a kinodynamically feasible trajectory. Another sampling-base path initialisation approach considering the kinodynamics of the UAV using STD-trees, is presented in [Ye et al., 2021a]. Overall sampling-based approaches tend to scale poorly to large, complex environments, especially in the presence of narrow gaps.

Graph search algorithms like A* [Hart et al., 1968] are used in numerous UAV trajectory optimisation frameworks to efficiently create an initial path [Nieuwenhuisen and Behnke, 2019, Ding et al., 2019]. The A* algorithm improves on the Dijkstra search algorithm by adding a conservative heuristic cost to the optimisation. Most implementations search for the shortest path to the goal position and use the distance between graph vertices as the edge cost. Based on this cost, the Euclidean distance to the goal position provides a conservative but accurate heuristic cost for the optimisation. [Liu et al., 2017b] and [Tordesillas et al., 2022] use a Jump Point Search (JPS) method to further accelerate the graph search, however this method requires a regular grid representation of the map. However for UAV motion planning the shortest path to the goal does not necessarily provide the fastest and safest path. In [Gao et al., 2018] a velocity field is used to compute the edge cost and improve the quality of trajectory initialisation. The hybrid-state A* search presented in [Dolgov et al., 2010] generates a kinodynamically feasible path for a 3D kinematic state space. In [Zhou et al., 2019] the search algorithm is extended to search for a feasible UAV trajectory initialisation. While search-based algorithms tend to scale poorly in higher dimensional search spaces, they reliably

find paths through narrow passages. To combine the best of all worlds, [Ye et al., 2022] proposes a kinodynamic sampling-based algorithm with region optimisation using local path segments computed with an A* search.

Another area of research focuses on the generation of homologically distinctive path initialisations. Correlating the geometric path to the quality of the optimised trajectory is difficult. Therefore optimising numerous topologically distinctive paths can help to find the best trajectory. In [Oleynikova et al., 2018] a sparse 3D topological graph is generated. The work extracts a 3D Generalised Voronoi Diagram (GVD) from a ESDF to initialise a skeleton diagram. Afterwards the diagram is converted into sparse graph and an initial path graph computed by running A* on sparse graph. Two similar topological path search algorithms are proposed in [Zhou et al., 2020] and [Ye et al., 2021b]. Both works use a sampling-based strategy. Once a sampled path segment interferes with an obstacle the approaches strategically place additional sample points around the obstacle to approximate the topology of the space. In [Penicka and Scaramuzza, 2022] a variant of Probabilistic Roadmap (PRM) is computed to find topological distinct paths in a hierarchical fashion.

One limitation of all previously mentioned approaches is that they do not take the UAV's orientation into account, only considering the position along the trajectory and assuming a spherical collision model. Since a sphere is not a good approximation for the shape of most UAVs, these approaches cannot plan paths through gaps or attitude constrained environments that the UAV otherwise can pass through.

2.2.6 Attitude Aware SE(3) Motion Planning

There are approaches to SE(3) planning through narrow gaps of negligible thickness. Using the relationship between the UAVs attitude and acceleration [Hirata

and Kumon, 2014] and [Loianno et al., 2017] navigate the UAV through gaps by incorporating handpicked attitude constraints at given spare positions into the path optimisation. In [Falanga et al., 2017] the autonomous gap detection is simplified by marking the gaps with a black-and-white rectangular pattern using a forward-facing camera. Once the gap is detected, a plane-constrained aligning with the gap is added to trajectory optimisation problem to make the gap traversal feasible. Alternatively, [Lin et al., 2019] introduces a deep learning framework, using a multi-layer perceptron to solve the problem of gap detection and trajectory generation with an end-to-end approach. The approach imitates from a traditional pipeline and is further optimised using reinforcement learning. However all above approaches focus on gap detection and attitude constrained trajectory generation rather than SE(3) trajectory planning.

A SE(3) search-based trajectory planning approach is proposed in [Liu et al., 2018] (see Fig. 2.5). Their work approximates the UAV's shape with an ellipsoid. Pre-computed trajectory motion primitives with known attitudes along the segment are used to construct the global trajectory and check for collisions. To increase the performance a lower dimensional search is conducted first to guide the higher dimensional trajectory sampling algorithm.

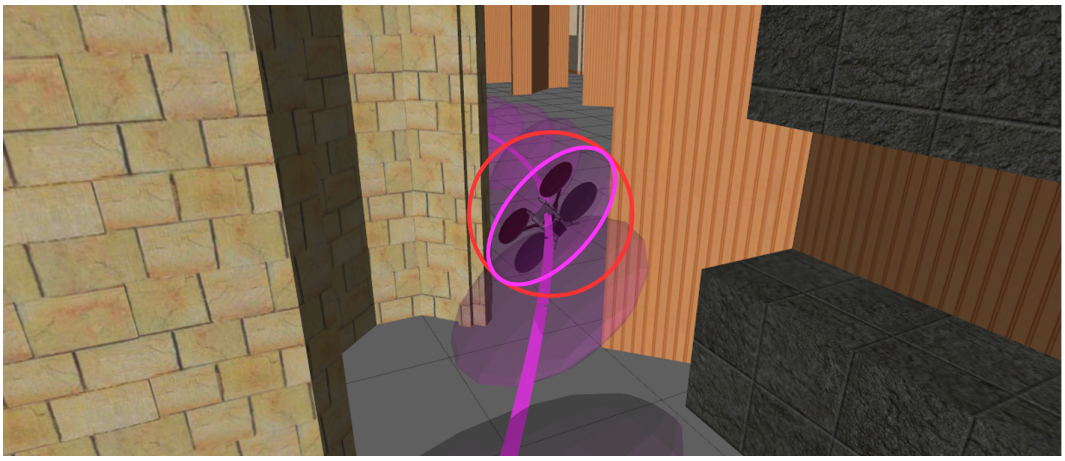

Figure 2.5: The SE(3) sampling-based planner proposed in [Liu et al., 2018] navigates the UAV through a narrow gap.

INSAT [Natarajan et al., 2021] generates SE(3) aware trajectories by interleaving a 5D discrete (position and attitude) weighted A* graph search algorithm with a trajectory generation stage. For each expanded 5D state a trajectory is optimised from the starting state to the end point's positional and attitude constraint. Once a trajectory to the goal state has been found all constraints are removed and the trajectory is further optimised until it is dynamically infeasible or in collision. The approach seems promising, but is computationally expensive and allows only limited optimisation of the final trajectory. On the contrary [Wang et al., 2022] and [Han et al., 2021] plan safe UAV trajectories in SE(3) using SFCs [Liu et al., 2017b] and a non-spherical convex shape approximation of the UAV. Rather than constraining the control points of a Bezier curve, their approach uses a standard polynomial representation. The polynomials are sampled in regular steps along the trajectory and the UAV orientations computed based on the relation to the acceleration. For each sample posed the UAV's convex shape approximation is constrained to stay within the SFC. While these methods plan trajectories in SE(3) in the optimisation stage, the initialisation of the SFC still does not consider the UAV's orientation, making them prone to fail unless manually selected waypoints are provided.

Similar to [Natarajan et al., 2021], we bridge the gap to fully autonomous SE(3) planning by introducing an efficient orientation aware path initialisation to generate the SFC and optimise the trajectory according to [Wang et al., 2022] and [Han et al., 2021].

2.2.7 Hierarchical Motion Planning

Hierarchical planning leverages hierarchical spatial data structures such as quadtrees and octrees to efficiently plan paths in large environments by using a graph search algorithm such as A* [Hart et al., 1968]. A hierarchical, multi-resolution version of A* was first used in [Kambhampati and Davis, 1986] to plan paths in a quadtree.

Since then, several methods leveraging similar ideas have appeared. For example in [Behnke, 2004], the grid resolution is decreased as the distance from the robot increases to allow efficient replanning. In [Lee and Yu, 2009], a systematic way to reduce the resolution of a 2D grid while preserving occupancy structure and a 2-stage, coarse-to-fine A* was proposed. In [Cowlagi and Tsiotras, 2012] a method for planning at fine resolution only in local neighborhood of the robot while taking dynamic constraints into account was presented. A method for iteratively generating local graphs from a hierarchical map representation was demonstrated in [Hauer et al., 2015] and improved in [Hauer and Tsiotras, 2016]. More recently, [Larsson et al., 2021] generates a coarse graph from a hierarchical map representation using information theoretic criteria. A common limitation of the aforementioned methods is their lack of consideration of the robot orientation in collision checks. Thus, they cannot plan paths through narrow gaps that the robot can fit through only in certain orientations – which our approach aims to address.

Chapter 3

Preliminaries

This chapter provides a summary of the notation and technical background that the research of this thesis is based on.

Contents of Chapter

3.1 Notation

The following notation has been used throughout the thesis.

$a \in \mathbb{R}$	A lower-case symbol denotes a scalar.
$\mathbf{a} \in \mathbb{R}^m$	A bold lower-case symbol denotes an m-dimensional column vector.
$\hat{\mathbf{a}} \in \mathbb{R}^{m+1}$	A dotted bold lower-case symbol denotes an $m+1$-dimensional homogeneous column vector $\dot{\mathbf{a}} = [a_0, a_1, \dots, a_m, 1]^T$.
$\mathbf{A} \in \mathbb{R}^{m \times n}$	A bold upper-case symbol denotes an $m \times n$ matrix.
${}_A\mathbf{a}$	A vector $\mathbf{a}$ expressed in coordinate frame A.
$\mathbf{T}_{AB} \in \mathrm{SE}(3)$	A Euclidean transformation from coordinate frame $\{B\}$ to coordinate frame $\{A\}$.
$\{W\}$	The World frame.
$\{C\}$	The Camera frame.
SO(3)	Special Orthogonal group: the group of 3D rotations.
SE(3)	Special Euclidean group: the group of 3d rigid transformations.
$\mathbf{u}_A$	2D pixel represented in the frame A.
s	A node's scale in the octree.
$\alpha, \beta, \gamma, \delta, \lambda$	All Greek letters are design parameters and defined in the parameter Table 5.2 in the evaluation section.

A node centre $\mathbf{x}$ corresponding to position $\mathbf{p}$ and with scale (octree level) s is denoted $\mathbf{x}(\mathbf{p}, s) \in \mathbb{R}^3$. Alternatively, we use an index i to access the nodes, corresponding to a tuple $(\mathbf{p}, s)$, e.g. centres as $\mathbf{x}_i$ or scales as s_i. The finest scale $s = 0$ will correspond to the finest resolution a_{finest} (in the order of centimeters).

3.2 Sensor Models

Our mapping framework supports both a basic pinhole camera model and a LiDAR sensor model. We provide a short description of the projection models of both sensors.

3.2.1 Pinhole Camera Model

We assume a standard pinhole camera model, mapping a 3D point ${}_W\mathbf{p}$ in homogeneous coordinates ${}_W\hat{\mathbf{p}} = [x, y, z, 1]^T$ in world frame to the 2D pixel coordinates $\mathbf{u}_I = (u, v)$. Given the transformation from world to camera frame, we express the point in camera frame

$$ {}_C\hat{\mathbf{p}} = \mathbf{T}_{CW}\,{}_W\hat{\mathbf{p}}. \tag{3.1} $$

Finally we project the point into the image plane according to

$$ \mathbf{u}_I = \mathbf{K}\left[\pi\left(\mathbf{T}_{CW}\,{}_W\hat{\mathbf{p}}\right)\right], \tag{3.2} $$

using the perspective projection function $\pi(.)$

$$ \pi\left({}_C\hat{\mathbf{p}}\right) = \frac{1}{z}\begin{bmatrix} x \\ y \end{bmatrix} \tag{3.3} $$

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

$$ \mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.4} $$

where f_u and f_v correspond to the camera's horizontal and vertical focal length and c_u and c_v to the horizontal and vertical offset to the optical axis.

3.2.2 LiDAR Sensor Model

The LiDAR model used in this thesis is adapted to an Ouster OS1-64. The sensor generates organised dense point clouds of 1024 x 64 points, with a vertical Field

of View (FoV) of 33.2°and a horizontal FoV of 360°. Each scan is converted to a spherical range image to be successfully integrated with our mapping framework. Given a 3D point ${}_C\mathbf{p} = [x, y, z]^T$ in camera frame, the azimuth ϕ and elevation ψ angle can be computed according to

$$\phi = \frac{360}{2\pi}\left(2\pi - \arctan\left(\frac{y}{x}\right)\right) \tag{3.5}$$

$$R = \sqrt{x^2 + y^2 + z^2} \tag{3.6}$$

$$\psi = \frac{360}{2\pi}\arcsin\left(\frac{z}{R}\right). \tag{3.7}$$

Considering the minimum and maximum elevation angles ψ_{min} and ψ_{max} respectively, the pixel coordinates of the range image correspond to

$$u = \frac{\phi - \phi_{\text{offset}}}{360} u_{\text{max}} \tag{3.8}$$

$$v = \frac{\psi - \psi_{\text{offset}}}{360} v_{\text{max}}. \tag{3.9}$$

Further adjustment to the horizontal image coordinate u might be needed to compensate for the azimuthal wrap-around.

3.3 *supereight* Single-res

A major advance in real-time dense mapping came with the introduction of KinectFusion [Newcombe et al., 2011]. KinectFusion represents the world as an implicit truncated-signed distance fields (TSDF) discretised in a regular three-dimensional grid. Nevertheless, KinectFusion scales poorly due to a densely pre-allocated single-resolution grid and is therefore not suitable for the exploration of large environments.

Octrees are a popular choice of hierarchical data structures as they make use of sparseness of the environment by only allocating dense structures where needed. Starting from a root node, covering the entire map, each node is recursively subdivided into eight nodes of equal size until a leaf node is reached. [Vespa et al.,

2018] developed an octree-based data structure, named *supereight*, which combines aggregated voxel blocks and a hierarchical indexing structure. Following the octree data structure, nodes are recursively subdivided and the parent-child relationship is established via pointer connectivity. Rather than storing a single voxel at leaf level, voxels are aggregated in 8^3 blocks and stored in an unstructured array. The template based data structure incorporates an underlying fusion, rendering and tracking pipeline, supporting TSDF and occupancy map representations. An illustration of the data structure is shown in Fig. 3.1. However, the implementation of *supereight* presented in [Vespa et al., 2018] only allows data representation at a single resolution for the entire map.

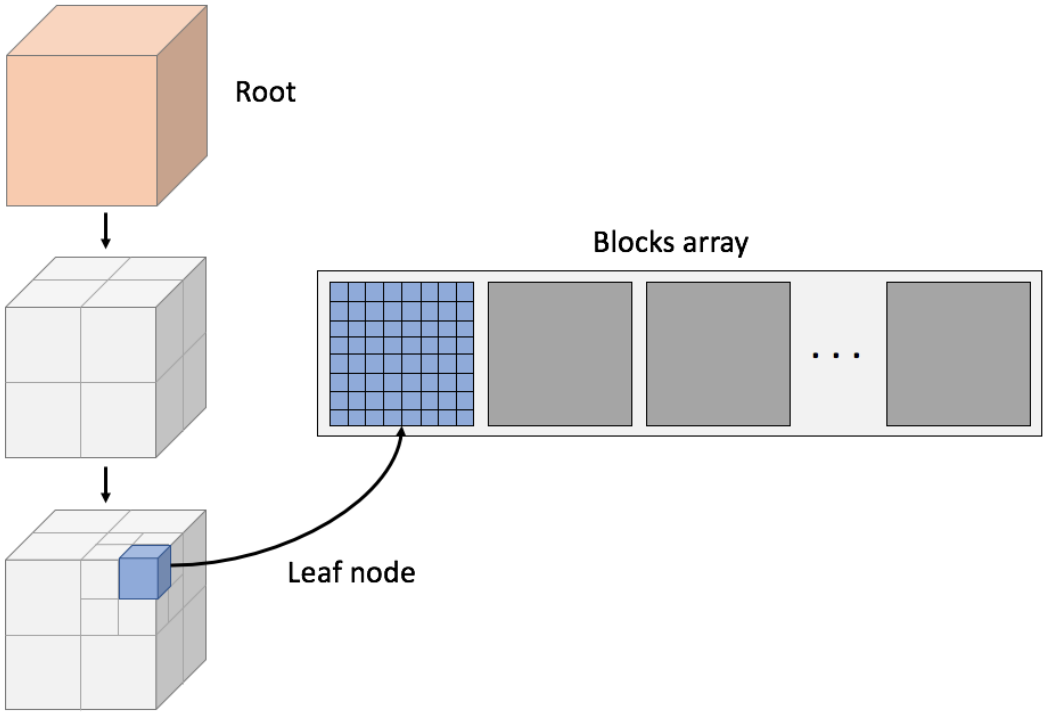

Figure 3.1: The recursive *supereight* octree data structure until leaf level (left). The blocks array containing all leaf level voxel blocks (right).

Depth estimation systems provide information at different scales depending on the measured distance, where far measurements cannot provide the same level of detail as close-up views. Consequently, using a single map resolution for the entire

spectrum of depth measurements causes aliasing effects [Vespa et al., 2019].

3.4 *supereight* Multi-res TSDF

While we do not claim the following as our own contribution, we took major part in the development, implementation and testing of the framework [Vespa et al., 2019].

[Vespa et al., 2019] introduces *Multi-res TSDF*, a modification of the SLAM framework presented in [Vespa et al., 2018], allowing TSDF integration at a adaptive resolution among individual blocks. Additionally, the changes include an adaptation of the dense tracking and mapping pipeline, such that the surface point cloud required for the Iterative Closest Point tracking algorithm is directly rendered from the adaptive resolution map. Overall, the framework improves aliasing effects and accelerates computation times.

Multi-res TSDF stores the values at an adaptive-resolution by adding mip-mapped layers to each block. The layers cover the range from the finest map scale to block scale and increase memory overhead by only 14%. The updated data structure is illustrated in Fig. 3.2

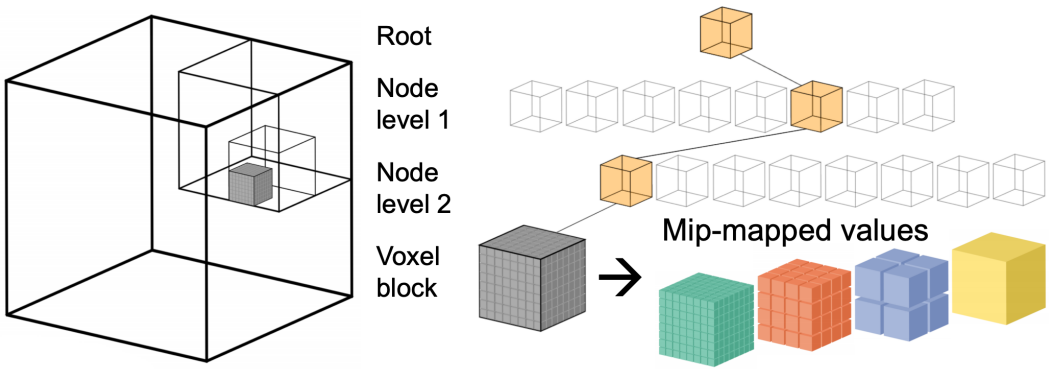

Figure 3.2: Illustration of the *supereight Multi-res TSDF* data structure (a) A single voxel block allocated at level 3 of the octree. (b) The decomposition of the octree, showing the pointer connectivity between nodes. Rather than only allocating the $8 \times 8 \times 8$ voxels at the finest scale of the block (green) [Vespa et al., 2018], additional mid-mapped levels (orange, blue, yellow) have been added to each block.

During the allocation stage, a ray-casting based strategy is used to add new voxel blocks in a truncated region $\pm\mu$ around the projection of the depth image. The integration scale of each voxel is kept equal for the whole block and based on the block's centre distance to the camera. To prevent aliasing effects, the voxel scale is chosen such that the back-projected voxel size and pixel size are as close as possible (see Fig. 3.3).

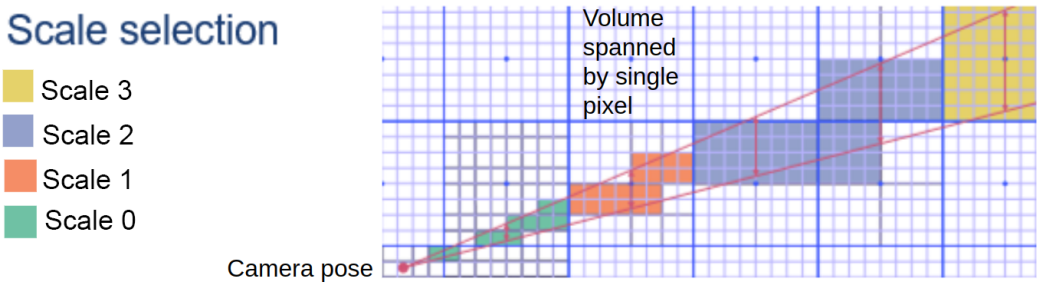

Figure 3.3: The chosen integration scale based on the distance to the camera.

New measurements are integrated into the map using weighted mean fusion and are based on the same inverse sensor model as in [Newcombe et al., 2011] and

[Vespa et al., 2018], while lazy propagation algorithms make sure the data structure is kept consistent between iterations.

After each depth frame integration, all voxel values at the current integration scale are up-propagated through the mid-mapped levels up to the block's scale (see Fig. 3.4 (a)). This is required to guarantee a common scale between neighbouring blocks to interpolate surface point and normals in the rendering stage. The framework shifts the cell's sample point from the node's corner to its centre such that the up-propagated parents value can be computed from the children's mean.

Contrary to the up-propagation, values are only down-propagated if a scale change to a finer scale is required. A scale change to a finer scale might happen if a surface moves closer to the camera than originally observed. It should also be mentioned that only unitary scale changes are allowed each iteration. If the voxel at the new scale has not been observed yet, the child value is computed using tri-linear interpolated of the neighbouring cells at the coarser scale. However, if the finer scale has been initialised previously a *delta down-propagation* approach similarly to [Zienkiewicz et al., 2016] is used. Rather than discarding the values at the finer scale, each child value is shifted according to the change of the parent value since the last up-propagation (see Fig 3.4 (c)).

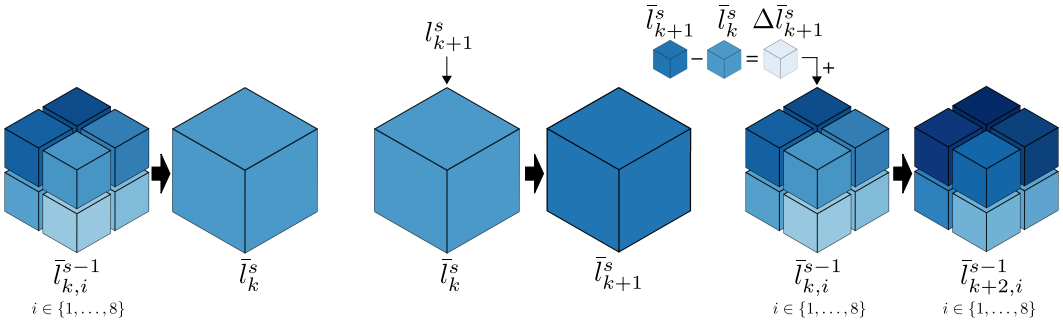

Figure 3.4: (a) Up-propagation: The parent value $\bar{l}^s_k$ at scale s is computed by computing the mean value of its eight children $\bar{l}^{s-1}_{k,i}$ at scale $s-1$, (b) Weighted mean integration of new measurements, (c) Delta down-propagation: Each child value $\bar{l}^{s-1}_{k,i}$ is shifted by the difference of the current parent value $\bar{l}^s_{k,i}$ and the previously up-propagated value $\bar{l}^s_{k,i}$.

3.5 A* Algorithm

The A* algorithm is an informed graph search algorithm aiming to find the smallest cost path to a goal location. Based on the Dijkstra's algorithm, the A* algorithm uses a directed heuristic function to a specified goal location to accelerate the search process.

To find the optimal path to the goal, it is required that the heuristic function $h(x)$ is conservative, underestimating the *cost-to-go*. Best results are archived with a heuristic function closest to the actual *cost-to-go*, while choosing $h(x) = 0$ renders the algorithm equivalent to the Dijkstra's algorithm. Additionally the algorithm requires a graph with positive edge cost $e(x_i, x_j) > 0$ connecting vertices x_i and x_j.

The algorithm keeps track of a *OPEN* list of vertices to be processed and a *CLOSED* list of already processed vertices. The *OPEN* list represents a priority queue sorting the vertices based on the lowest cost of the evaluation function

$$f(x_i) = g(x_i) + h(x_i), \tag{3.10}$$

where $g(x_i)$ is the currently best cost to reach x_i from the start vertex, and $h(x_i)$ the heuristic cost to the goal vertex.

To start the algorithm the *OPEN* list is initialised with the start vertex x_{start}. From there the algorithm proceeds iteratively until the goal node is reached. During each iteration the vertex with the lowest cost $f(x_i)$ is removed from *OPEN* and added to *CLOSED*. The costs of the vertex successors $x_{s,j}$ are computed according to

$$g(x_{s,j}) = g(x_i) + e(x_i, x_{s,j}) \tag{3.11}$$

and the algorithm terminates if the vertex corresponds to the goal. If the newly computed cost of the successor is lower than its current, the successor is added to *OPEN* or discarded otherwise. Before the vertex is added to *OPEN*, the vertex's cost and predecessor needs to be updated. Additionally it is required to first remove the vertex from either *OPEN* or *CLOSED*, if applicable.

The full algorithm is presented in Alg. 1.

Algorithm 1 A* Algorithm

```
Initialise OPEN list
Initialise CLOSED list
Add x_start to OPEN
while OPEN not empty do
    Get node x with smallest cost f(x) from OPEN list
    Add node x to CLOSED list
    if x contains x_goal then
        Lowest cost path found → Terminate
    for all Successors x_s of x do
        Compute cost to reach g(x_s) based on g(x)
        Compute heuristic cost h(x_s)
        Compute evaluation cost f(x_s)
        if x_s in OPEN ∧ new cost not lower than existing one then
            Discard node and continue
        else if x_s in CLOSED ∧ new cost not lower than existing one then
            Discard node and continue
        Remove x_s from OPEN or CLOSED list if applicable
        Make x predecessor of x_s
        Update cost of x_s
        Add x_s to OPEN list
```

3.6 ESDF Wave-Front Propagation

[Lau et al., 2010] presents a brushfire method to compute ESDF maps from occupancy maps. Our adaptive-resolution ESDF computation in Chapter 5 is a modification of their method. While their work focuses on incremental updates including appearing and disappearing obstacles, we present a simplified version of their algorithm for known environments, not considering the removal of objects.

During the first stage all occupied cells are initialised with an ESDF value of 0, referring to themselves as the closest obstacle. The ESDF value of all remaining voxels is set to infinity. To compute a truncated ESDF as in [Oleynikova et al., 2017] an initial ESDF value equivalent to the ESDF boundary should be chosen.

The algorithm shows close resemblance to the A* algorithm introduced before. All occupied cells are added to the *OPEN* priority queue and sorted according to the shortest ESDF distance. During each iteration the cell x with the shortest distance is taken from *OPEN* and the distance of its neighbour cells $x_{s,i}$ computed based on the closet obstacle of x. If the computed distance lowers the ESDF value of $x_{s,i}$, its ESDF value is update and the cell stores the reference to the same closest obstacle as x. Otherwise the cell is discarted. [Han et al., 2019] shows in FIESTA that choosing the 6-connectivity instead of the 26-connectivity only marginally effects the ESDF accuracy while significantly improving computation times. All updated cells are added to the priority queue and once *OPEN* is empty, the algorithm terminates. The full wave-front propagation algorithm is outlined in Alg. 2.

Algorithm 2 ESDF Wave-Front Propagation

```
1:  InitialiseMap()
2:  for all x in MAP do of
3:      if isOcc(x) then
4:          obst(x) ← x
5:          dist(x) ← 0
6:          Add x to OPEN list
7:      else
8:          dist(x) ← ∞
9:
10: WaveFrontPropagation()
11: while OPEN not empty do
12:     Get cell with smallest distance d(x) from OPEN list
13:     for all Neighbours x_s of x do
14:         Compute dist_tmp based on obst(x)
15:         if dist_tmp(x_s) < dist(x_s) then
16:             Discard cell and continue
17:         dist(x_s) ← dist_tmp
18:         obst(x_s) ← obst(x)
19:         Add x_s to OPEN list
```

3.7 Safe Flight Corridor Generation

A Safe Flight Corridor (SFC) provides a locally convex free-space approximation by concatenating a set of overlapping convex shape priors. The polyhedron SFC presented [Liu2017a] has proven to be an advantageous approximation as it provides a flexible and accurate free-space approximation that can be described in simple mathematical terms requiring only linear inequality constraints.

Given a piece-wise linear path with M line segments S_j a polyhedron C_j is computed for each segment, where consecutively overlapping polyhedra guarantee the continuity of the SFC. The polyhedron generation procedure of each segment is split into two parts, (1) "Find Ellipsoid" and (2) "Find Polyhedron".

To find an obstacle-free ellipsoid containing the line segment S_j a sphere with a diameter equal to the segment length is placed at the segment's centre. Next the sphere is iteratively shrunk to the largest spheroid that does not contain any obstacles. Lastly the spheroid is stretched along a third axis to find the maximum initial ellipsoid ξ_j^0.

In a subsequent step the ellipsoid is used to find the hyper planes of the polyhedron. Given the initial intersection of the ellipsoid with the environment a tangent half space $H_0 = \{\mathbf{p}|\mathbf{D}^T\mathbf{p} < \mathbf{d}\}$ is places in the collision point. All obstacles outside the half space are removed and the ellipsoid is expanded, with constant aspect ratio, until in collision occurs to compute the next half space H_1. The process is repeated until the polyhedron $C_j = \cap_{k=1}^{N_j} H_k$ constructed of N_j half spaces does not contain any obstacles anymore.

To reduce the computational burden of the corridor construction only the local obstacles within a bounding box (BB) around each line segment are considered. Therefore the six hyper planes of each BB provide the initial half spaces of the corresponding polyhedron. The full process is further illustrated in Fig. 3.5. As

expected the position of the waypoints significantly influences the shape of the polyhedron. Contrary to state-of-the-art approaches we are taking this property into account during the corridor construction.

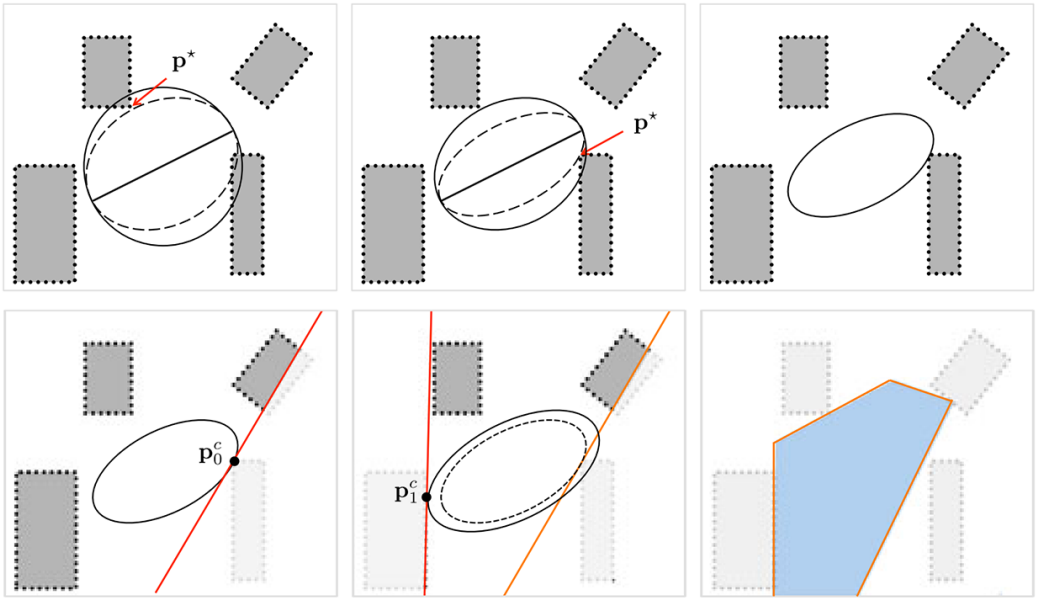

Figure 3.5: Polyhedron free-space approximation: (top) Ellipsoid computation: A sphere, centred around the line segment, is iteratively shrunk to the largest obstacle free spheroid. Lastly the spheroid is expanded along a third axis to find the maximum ellipsoid. (bottom) Polyhedron computation: A tangential half space is placed in the point of collision and all obstacles outside the half space are removed. The ellipsoid is inflated to the next point of collision, maintaining its aspect ratio. The steps of inflation and constraint addition is iteratively repeated until no obstacles remain in the union of half space constraints (All figures taken from [Liu et al., 2017b]).

3.8 Trajectory Optimisation

This section provides a description of the optimisation methods used as the final steps of our overall motion planning pipelines. While this section is not required to understand the contribution of this thesis, it is provided for overall completeness. Further details on the Bezier curve optimisation in Safe Flight Corridors can be

found in [Gao et al., 2018]. Additionally, the used SE(3) trajectory optimisation method is extensively described in [Wang et al., 2022].

3.8.1 Differential Flatness & Polynomial Motion Primitive

In [Mellinger and Kumar, 2011] it was shown that the state and inputs of the UAV can be expressed by the four flat outputs $\sigma(t) = [\mathbf{p}(t), \psi(t)]^T$ and their derivatives, where $\mathbf{p}(t)$ corresponds to the position of the UAV's centre of mass and $\psi(t)$ to its independent yaw angle. Based on the differential flatness property of the UAV model the trajectory has to be continuous up to snap (C^4) to be dynamically feasible for a UAV. Therefore, a common practice is to generate a spline trajectory out of multiple continuous time-parameterised polynomials of order $n \geq 4$ that are executed in consecutive order. For the entire spline to be smooth, the end derivatives of adjacent segments have to be equivalent. An example spline is illustrated in Fig. 3.6.

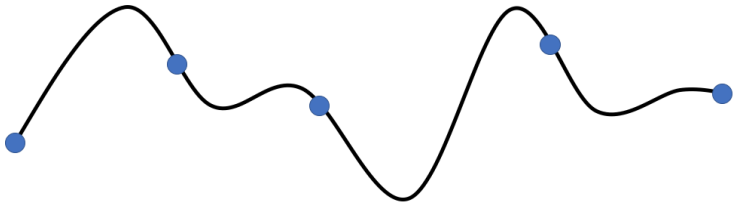

Figure 3.6: An example spline trajectory built out of four polynomial segments. The spline is constrained by the start and end position as well as three way-points (blue).

All trajectory segments S_j, with $j \in \{1, \dots, M\}$ can be described by a vector $\mathbf{p}_j(t)$ containing three n-th order polynomials $p_{\mu,j}(t)$, with $\mu \in \{x, y, z\}$. Each of the three polynomials represents the time dependent development of a single dimension of the UAV position.

$$\mathbf{p}_j(t) = \sum_{i=0}^{n} \mathbf{a}_{j,i} t^i \qquad \mathbf{a}_{j,i} = \left[a_{x,j,i}, a_{y,j,i}, a_{z,j,i}\right]^T \in \mathbb{R}^3, \tag{3.12}$$

An alternative way to express the evolution of the UAV position is by the vector product of $\tau(t)$, containing the powers of the time and the polynomial coefficient matrix $\mathbf{A}_j$ holding the polynomial coefficients $a_{\mu,j,i}$, with $i \in \{0,\dots,n\}$ for each dimension.

$$\mathbf{p}_j(t)^T = \mathbf{A}_j^T \tau(t), \quad t \in [0, T_j] \qquad \tau(t) = \begin{bmatrix} 1 & t^1 & t^2 & \dots & t^n \end{bmatrix} \tag{3.13}$$

A common practice is to minimise the squared jerk or snap derivative of the spline ($h = 3$ or $h = 4$), to reduce either camera shaking or energy consumption along the trajectory. Summed over all dimensions and trajectory segments the optimisation can be summarised as follows:

$$\min J_d = \min \sum_{\mu \in \{x,y,z\}} \sum_{j=1}^{M} \int_0^{T_j} p_{\mu,j}^{(h)}(t)^2 dt, \tag{3.14}$$

$$\text{s.t.} \quad \frac{d^l \mathbf{p}_{\mu,j}(t)}{dt^l}\Big|_{t=T_j} = \frac{d^l \mathbf{p}_{\mu,j+1}(t)}{dt^l}\Big|_{t=0}. \tag{3.15}$$

The differentiation of the polynomials provides a linear mapping of the polynomial coefficients to the end derivatives

$$\mathbf{M}_j \mathbf{a}_{\mu,j} = \mathbf{d}_{\mu,j} \tag{3.16}$$

Consequently the polynomial coefficients can be expressed in terms of the end-derivates by inverting the mapping matrix $\mathbf{M}_j$.

$$\mathbf{a}_{\mu,j} = \mathbf{M}_j^{-1}\mathbf{d}_{\mu,j}, \tag{3.17}$$

By combining the matrix representations in Equation 3.13 and 3.17 the initial objective function can be reformulated into an unconstrained quadratic program (QP):

$$\begin{aligned}
\min J_d &= \min_{d_F} \sum_{\mu\in\{x,y,z\}} \sum_{j=1}^{M} \int_0^{T_j} p_{\mu,j}^{(h)}(t)^2 dt \\
&= \min_{d_F} \bar{\mathbf{d}}^T \mathbf{O} \bar{\mathbf{d}} = \min_{d_F} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^T \begin{bmatrix} \mathbf{O}_{FF} & \mathbf{O}_{FP} \\ \mathbf{O}_{FP}^T & \mathbf{O}_{PP} \end{bmatrix} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} \\
&= \min_{d_F} \mathbf{d}_F^T \mathbf{O}_{FF} \mathbf{d}_F + 2\mathbf{d}_F^T \mathbf{O}_{FP} \mathbf{d}_P,
\end{aligned} \tag{3.18}$$

where $\mathbf{O}$ includes a reorganisation of the end-derivatives into free $\mathbf{d}_F$ and fixed $\mathbf{d}_P$ derivatives.

3.8.2 $\mathbb{R}^3$ Trajectory Optimisation in SFC

Every n-th order polynomial can be described by a Bernstein polynomial on the interval $t \in [0, T_j]$. An n-th order Bernstein polynomial is the weighted sum of n Bernstein bases $b_{j,i}(t)$, while each Bernstein basis is an n-th order polynomial by itself. Therefore a Bernstein polynomial is a superposition of n polynomials weighted by the control points $\mathbf{c}_{j,i}$:

$$\begin{aligned}
\mathbf{p}_j(t) = \mathbf{c}_{j,n} \left(\frac{t}{T_j}\right)^n + \mathbf{c}_{j,n-1} \left(\frac{t}{T_j}\right)^{n-1} \left(1 - \frac{t}{T_j}\right) + \ldots \\
+ \mathbf{c}_{j,n} \left(\frac{t}{T_j}\right) \left(1 - \frac{t}{T_j}\right)^{n-1} + \left(1 - \frac{t}{T_j}\right)^n
\end{aligned} \tag{3.19}$$

$$= \sum_{i=0}^{n} \mathbf{c}_{j,i} b_{j,i}(t), \quad \text{for} \quad t \in \left[0, T_j\right],$$

with

$$\mathbf{c}_{j,i} = \left[c_{x,j,i}, c_{y,j,i}, c_{z,j,i}\right]^T \in \mathbb{R}^3 \qquad b_{j,i}(t) = \binom{n}{i} \left(\frac{t}{T_j}\right)^i \left(1 - \frac{t}{T_j}\right)^{n-i}. \tag{3.20}$$

The advantage of using a Bernstein polynomial representation is that the control points build a convex hull that constrains the polynomial. Thus, by constraining the control points to stay within the SFC, the trajectory is guaranteed to stay within the SFC at all times as well. Fig. 3.7 illustrates a 4-th order Bernstein polynomial and the convex hull built by the five control points.

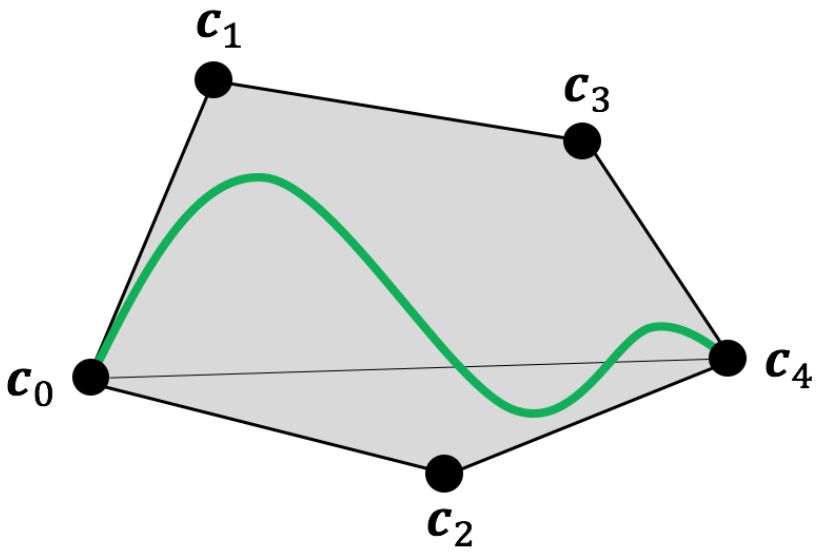

Figure 3.7: 4-th order Bernstein polynomial. The five control points build a convex hull that the polynomial is guaranteed to stay within.

[Gao et al., 2018] showed in their work the relation between the end-point derivatives and the control points using an auxiliary variable. Using this representation,

we can directly map the end-point derivatives to the control points of all segments with the mapping matrix $\mathbf{E}$

$$\bar{\mathbf{c}} = \mathbf{E}^{-1}\bar{\mathbf{d}}, \tag{3.21}$$

Therefore collision avoidance of the trajectory can be incorporated into the optimisation in Equation 3.18, by adding the inequality constraints of the control points to the QP as follows:

$$\min J_d = \bar{\mathbf{d}}^T \mathbf{O}\bar{\mathbf{d}} \tag{3.22}$$

$$s.t. \quad \mathbf{B}_{ie}\mathbf{E}^{-1}\bar{\mathbf{d}} \leq \mathbf{b}_{ie}. \tag{3.23}$$

3.8.3 SE(3) Trajectory Optimisation in SFC

[Wang et al., 2022] and [Han et al., 2021] present a SE(3) UAV trajectory planning framework to navigate attitude constrained environments containing narrow openings or tilted gaps. Instead of relying on boundary conditions, the approach provides a first solution to full SE(3) trajectory optimisation. On a high level the method constraints a convex shape approximation of the UAV to stay inside the SFC, considering its attitude.

Based on the differential flatness property, the UAV's attitude $\mathbf{R}_{WB} = [\mathbf{r}_{bx}, \mathbf{r}_{bz}, \mathbf{r}_{bz}]$ can directly linked to the acceleration along the trajectory as follows:

$$\mathbf{f}_b = \ddot{\mathbf{p}} + g_w\mathbf{e}_3, \tag{3.24}$$

$$\mathbf{r}_{bz} = \frac{\mathbf{f}_b}{\|\mathbf{f}_b\|_2}, \tag{3.25}$$

$$\mathbf{r}_{bz} = \frac{\mathbf{f}_b}{\|\mathbf{f}_b\|_2}, \tag{3.26}$$

$$\mathbf{r}_{ix} = [\cos\psi, \sin\psi, 0]^T, \tag{3.27}$$

$$\mathbf{r}_{by} = \frac{\mathbf{r}_{bz} \times \mathbf{r}_{ix}}{\|\mathbf{r}_{bz} \times \mathbf{r}_{ix}\|_2}, \tag{3.28}$$

$$\mathbf{r}_{bx} = \mathbf{r}_{by} \times \mathbf{r}_{bz}, \tag{3.29}$$

where g_w represents the gravitational acceleration along $\mathbf{e}_3 = [0, 0, 1]^T$.

[Han et al., 2021] models the UAV shape by a convex polyhedron, where the i-th vertex $\bar{\mathbf{v}}^i$ of the bounding volume, with offset ${}_B\mathbf{q}_v^i$, at time t is computed as as

$$\bar{\mathbf{v}}^i(t) = \mathbf{R}_{WB}(t)_B q_v^i + {}_W\mathbf{x}(t) \qquad \forall i \in \{1, 2, \dots, K\}. \tag{3.30}$$

Furthermore, the SFC is build from a series of M overlapping convex polyhedra $C_j \quad \forall j \in \{1, 2, \dots, M\}$, where each closed polyhedron is represented by N_j hyperplanes with normal vectors $\mathbf{n}_j^k$ and points $\mathbf{p}_j^k$ on the plane $\forall k \in \{1, 2, \dots, N_j\}$.

The overall optimisation is comprised of the sum of a smoothness term, time parameterisation, dynamic feasibility cost for velocity and acceleration as well collision cost, penalising the crossing of the SFC. In summary the optimisation is formulated as:

$$\begin{aligned}
\min_{\mathbf{u}_j(t), \forall j} H = \Sigma_{j=1}^{M} \int_{T_{j-1}}^{T_j} \mathbf{u}_j(t)^T \mathbf{u}_j(t) dt + \rho(T_M - T_0) + \\
W_v \Sigma_{j=1}^{M} \int_{T_{j-1}}^{T_j} \mathcal{K}\left(\|\mathbf{p}_j^{(1)}(t)\|_2^2 - v_{\max}^2\right) dt + \\
W_a \Sigma_{j=1}^{M} \int_{T_{j-1}}^{T_j} \mathcal{K}\left(\|\mathbf{p}_j^{(2)}(t)\|_2^2 - a_{\max}^2\right) dt + \\
W_c \Sigma_{j=1}^{M} \int_{T_{j-1}}^{T_j} \Sigma_{i=1}^{K} \Sigma_{k=1}^{N_j} \mathcal{K}\left((\bar{\mathbf{v}}^i(t) - \tilde{\mathbf{p}}_j^k)^T \vec{\mathbf{n}}_j^k\right) dt
\end{aligned} \tag{3.31}$$

$$s.t. \quad \mathbf{u}_j(t) = \mathbf{p}_j^{(h)}(t), \quad \forall t \in [T_{j-1}, T_j], \tag{3.32}$$

$$\mathbf{p}_j^{[d]}(T_j) = \mathbf{p}_{j+1}^{[d]}(0), \mathbf{p}_j^{\tilde{d}}(T_j) = \bar{\mathbf{p}}_j, \tag{3.33}$$

$$\mathbf{p}_j(T_j) \in \mathcal{C}_j \cap \mathcal{C}_{j+1}, \quad T_j - T_{j-1} > 0, \tag{3.34}$$

$$\forall j \in \{1, \dots, M\}, \tag{3.35}$$

$$\mathbf{p}_1^{[h-1]}(0) = \bar{\mathbf{p}}_0, \quad \mathbf{p}_M^{[h-1]}(T_M) = \bar{\mathbf{p}}_f, \tag{3.36}$$

where $\mathcal{K}(x) = max(x, 0)^3$ is a cubic penalty, ρ is the weight of the regularisation and W_v, W_a, W_c are the weights for the dynamic feasibility and collision cost respectively. Furthermore, $\bar{\mathbf{p}}_0$ and $\bar{\mathbf{p}}_f$ represents the initial and final state, while $\bar{\mathbf{p}}_j^{[d]}$ corresponds to the interval condition.

Due to the polynomial motion primitive, the objective function is a function of all polynomial coefficients $\mathbf{A}$ and segment times $\mathbf{T}$

$$H = H(\mathbf{A}, \mathbf{T}). \tag{3.37}$$

with partial derivatives

$$\frac{\partial H(\mathbf{A}, \mathbf{T})}{\partial \mathbf{A}}, \tag{3.38}$$

and

$$\frac{\partial H(\mathbf{A}, \mathbf{T})}{\partial \mathbf{T}}. \tag{3.39}$$

The derivative of the smoothness term can be easily computed given that it is a polynomial composed of $\mathbf{A}$ and $\mathbf{T}$. However the partial derivative of the dynamic constraint term and collision cost is more complex and can be simplified by the discretised approximation of the penalty terms using L sampled constraint points along the trajectory. Further details regarding the computation of the partial derivatives can be found in [Han et al., 2021]. Furthermore, $\mathbf{A}$ and $\mathbf{T}$ can be related to the waypoints $\mathbf{q} = (q_1, \dots, q_{M-1}) \in \mathbb{R}^{3\times M-1}$ and the boundary states by a non-singular mapping matrix, such that the optimisation can be reformulated as:

$$\min \tilde{H}(\mathbf{q}, \mathbf{T}) = H(\mathbf{A}(\mathbf{q}, \mathbf{T}), \mathbf{T}), \tag{3.40}$$

$$s.t. \quad \mathbf{q}_j \in \mathcal{C}_j \cap \mathcal{C}_{j+1}, \quad \forall j \in \{1, \dots, M-1\} \tag{3.41}$$

$$T_j - T_{j-1} > 0 \quad \forall j \in \{1, 2, \dots, M\}. \tag{3.42}$$

with the partial derivatives

$$\frac{\partial \tilde{H}(\mathbf{q}, \mathbf{T})}{\partial \mathbf{q}} = \frac{\partial H}{\partial \mathbf{A}} \frac{\partial \mathbf{A}}{\partial \mathbf{d}}, \tag{3.43}$$

and

$$\frac{\partial \tilde{H}(\mathbf{q}, \mathbf{T})}{\partial \mathbf{T}} = \frac{\partial H}{\partial \mathbf{T}} + \frac{\partial H}{\partial \mathbf{A}} \frac{\partial \mathbf{A}}{\partial \mathbf{d}}. \tag{3.44}$$

Following [Wang et al., 2022], the temporal and spatial constraints can be removed and the optimisation robustly solved with a quasi-Newton method. In contrast to $\mathbb{R}^3$ SFC based methods the approach requires significantly more constraints and additional gradient computation, making it computationally more expensive, however this can be compensated for by computing the gradients in parallel [Han et al., 2021]. Nevertheless, the approach still finds the global minimum solution for the given SFC independent of the initialisation of the polynomial coefficients.

Chapter 4

Multi-resolution Occupancy Mapping

Contents of Chapter

Parts of this Chapter appear in:

Nils Funk, Juan Tarrio, Sotiris Papatheodorou, Marija Popović, Pablo F. Alcantarilla and Stefan Leutenegger. **Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning**. *IEEE Robotics and Automation Letters (RA-L).* Vol. 6, No. 2, pp. 3553-3560, 2021. [Funk et al., 2021]

Yiduo Wang, Nils Funk, Milad Ramezani, Sotiris Papatheodorou, Marija Popović, Marco Camurri, Stefan Leutenegger and Maurice Fallon. **Elastic and Efficient LiDAR Reconstruction for Large-Scale Exploration Tasks**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2021.* [Wang et al., 2021a]

4.1 Introduction

In this chapter we introduce a volumetric adaptive-resolution *dense* mapping framework that supports multi-resolution queries and data integration using occupancy mapping [Hornung et al., 2013, Elfes, 1989, Thrun, 2003]. Contrary to methods based on signed distance functions, we continuously maintain a high resolution 3D octree representation of observed occupied and *free* space in real-time.

Our key insight is to recognise the lack of a concise method for defining required resolution in the context of occupancy mapping, with few approaches tackling this problem by extending single resolution approaches with ad-hoc heuristics [Vespa et al., 2018]. As a result, we designed a novel integration algorithm that selects resolution by constraining the induced sampling error in the observed occupancy, splitting an octree in a coarse-to-fine fashion until a desired accuracy is reached. Central to this idea is the introduction of a multi-scale max-min pooling of the input depth image which enables real-time operation by providing a conservative indication of measured depth variation in any given volume with only a few queries.

To further enhance performance, we modified the data structure of [Vespa et al., 2019] which uses mip-mapped voxel blocks, and carefully designed a new scale selection and data propagation scheme between levels. This allowed us to increase computational and memory efficiency without introducing reconstruction artefacts.

While our method focuses on RGB-D mapping, it is flexible to different sensor modalities. This is shown by evaluating our system in a wide range of datasets, from synthetic to large scale LiDAR, showing significant improvements in reconstruction accuracy, runtime performance, and planning performance against state-of-the-art approaches.

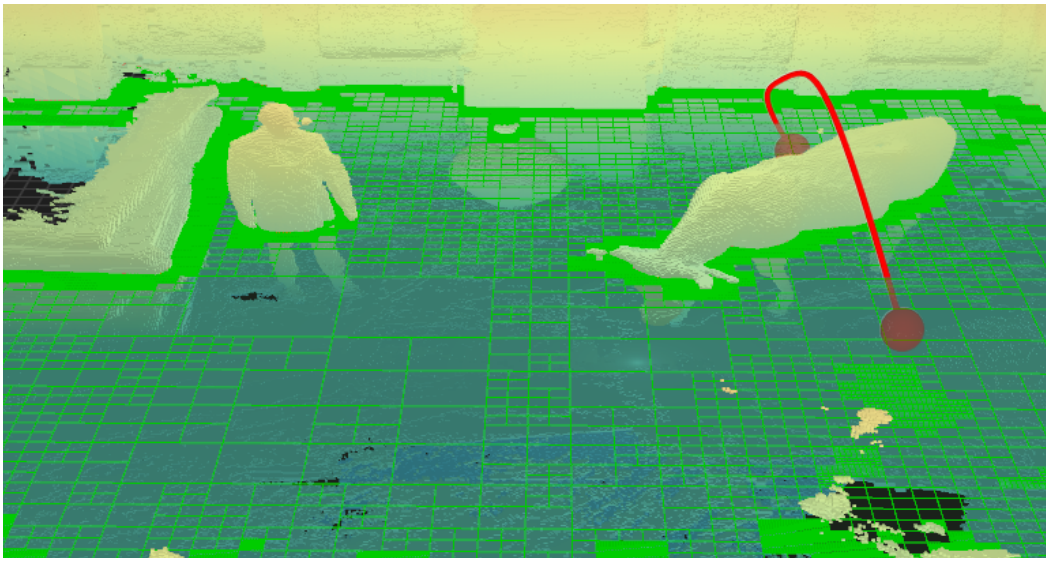

Figure 4.1: The main goal of our system is to provide a navigable 3D occupancy map with a defined notion of observed free space in small to large scale scenarios, using adaptive resolution to constrain memory consumption. The figure shows the output of our mapping pipeline for the *Cow and Lady* RGB-D dataset [Oleynikova et al., 2017] alongside a planned trajectory (red). The chosen voxel resolutions for free space encoding are shown for a map slice.

In summary, the main contributions are:

1. A dense volumetric multi-resolution system, comprising a data structure, fusion method and sensor models which together enable real-time online probabilistic occupancy mapping and accurate surface reconstruction, by consistently representing free and occupied space to fine resolutions where needed.

2. A novel fast map-to-image allocation algorithm and integration method, that seamlessly adapts the map to different scene scales and sensor modalities.

3. A comprehensive evaluation with respect to the state-of-the-art revealing vast improvements in the trade-offs of mapping accuracy, speed, memory consumption, tracking accuracy, and planning performance.

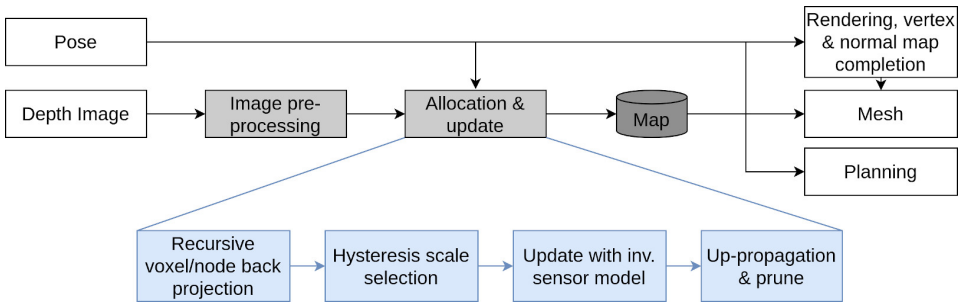

Figure 4.2: Overview of our system: The different stages of the mapping pipeline are shown in grey, while the blue boxes show the steps in the *allocation and updating* procedure. The flow of information is illustrated by the arrows. The allocation and updating stages modify the map, while the rendering and planning stages utilise the map information.

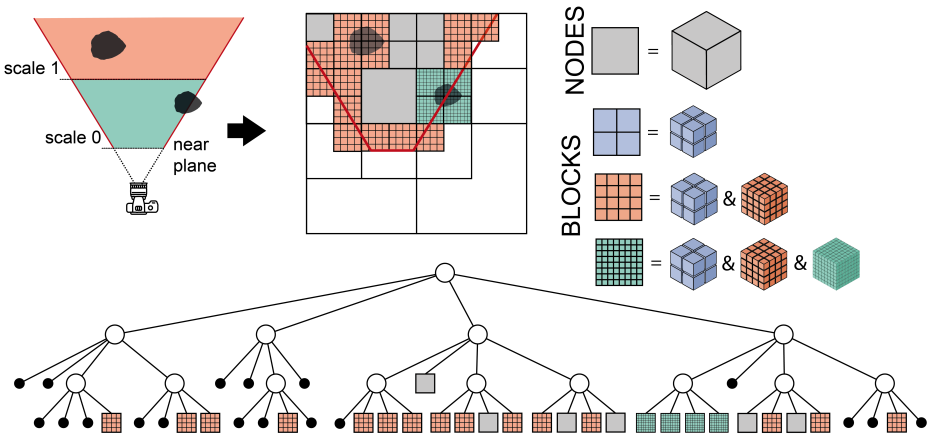

Figure 4.3: Overview of the data structure for a hypothetical 2D case for a pinhole camera model. Data is represented in a two tier octree having voxel blocks in the lower levels. Depending on the distance based integration scale s_c voxel blocks containing data up to different resolutions are allocated, resulting in important memory and time savings for cases where higher resolution is not needed. Note that in the given example the integration scale s_c for blocks only containing free and unknown space has a lower bound of $s_c > 1$ (Section 4.2.4).

4.2 Multi-Resolution Occupancy Mapping

Our library takes as input depth information and poses, incrementally computing a map which can be queried at any time for path planning, meshing, and rendering (see Fig. 4.2). We also provide an Iterative Closest Point (ICP) module that can be switched on to obtain a full Simultaneous Localisation and Mapping (SLAM)

system in the spirit of [Newcombe et al., 2011], as shown in Section 4.3.4.

In order to represent an occupancy efficiently using adaptive resolution we use an octree where, similar to [Vespa et al., 2018], the last levels consist of densely allocated voxel blocks aggregating up to $8 \times 8 \times 8$ voxels at the finest scale. This two tier memory structure leverages flexibility in choosing resolution at the higher (octree) levels with efficient access at lower (voxel block) level. Taking inspiration from [Vespa et al., 2019], each voxel block stores a pyramid representation of itself enabling fast occupancy updates at different resolutions. However, unlike [Vespa et al., 2019], we save memory by not keeping all mipmap levels and only allocating data in each block down to a single dynamically changing integration scale s_c, in other words our design has different voxel block types which are changed dynamically depending on the needed integration scale. More importantly, we augment this method by also allowing data to be stored at node level when the voxel block resolution is finer than needed. A representation of this structure can be seen in Fig. 4.3 for a hypothetical 2D case.

For a given point, the relevant occupancy information is maintained only at the lowest allocated voxel that contains it. Similar to [Hornung et al., 2013], upper non-leaf nodes store a max pooling of the children occupancy, enabling fast conservative queries of any given size. In addition to the max occupancy, we keep a Boolean state indicating whether unknown data is present in the children. This allows us to disambiguate a node as being partially or fully unobserved. In our system this pooling, or data up-propagation, is computed in each integration step, making it available for online planning at any given time.

4.2.1 Occupancy Map Fusion

Given a depth image $\mathbf{D}_k \in \mathbb{R}^{H \times W}$ and camera pose $\mathbf{T}_{WC_k}$ at time step k, the probability that a point ${}_W\mathbf{p}$ in 3D space is occupied is assumed to depend on the distance to the

camera and the corresponding depth measurement along the ray from the camera centre ${}_W\mathbf{c}$ to ${}_W\mathbf{p}$. Given the depth measurement z from the projection of point ${}_W\mathbf{p}$ into the camera, we represent the occupancy probabilities of one depth image in 3D:

$$P_{\text{occ}}\left({}_W\mathbf{p} \,|\, \mathbf{D}_k, \mathbf{T}_{WC_k}\right) = P_{\text{occ}}\left({}_W\mathbf{p} \,|\, z = \mathbf{D}_k\left[\pi(\mathbf{T}_{WC_k}^{-1}\,{}_W\mathbf{p})\right]\right), \tag{4.1}$$

which corresponds to the inverse sensor model [Thrun, 2003], a function of the depth along the ray. For brevity, it is referred to as $P_{\text{occ}}({}_W\mathbf{p}|z)$ in the following descriptions. An alternative way of representing the occupancy probability is using log-odds, which allows for Bayesian updates to be additive as:

$$l_k({}_W\mathbf{p}) = \log \frac{P_{\text{occ}}({}_W\mathbf{p}|z)}{1 - P_{\text{occ}}({}_W\mathbf{p}|z)}, \tag{4.2}$$

$$L_k({}_W\mathbf{p}) = L_{k-1}({}_W\mathbf{p}) + l_k({}_W\mathbf{p}). \tag{4.3}$$

In contrast to previous work, we do not accumulate the log-odds in a single sum, but instead use a weighted mean $\bar{L}_k({}_W\mathbf{p})$, with weight w_k, similar to how it is done in [Newcombe et al., 2011] providing additional information about the number of integrations. Thus the mean log-odds is updated as:

$$\bar{L}_k({}_W\mathbf{p}) = \frac{\bar{L}_{k-1}({}_W\mathbf{p})\, w_{k-1} + l_k({}_W\mathbf{p})}{w_{k-1} + 1}, \tag{4.4}$$

$$w_k = \min\{w_{k-1} + 1, w_{\max}\}, \tag{4.5}$$

while the accumulated log-odds can be preserved:

$$L_k({}_W\mathbf{p}) = \bar{L}_k({}_W\mathbf{p})\, w_k. \tag{4.6}$$

By clamping the weight w_k below a threshold $w_{\max}$, the influence of outliers and dynamic objects is mitigated. Moreover, using a maximum weight $w_{\max}$ rather than an occupancy threshold prevents the occupancy field at the surface from converging to a step function and does not require the use of a time based windowed updating step as in [Vespa et al., 2018].

4.2.2 Inverse Sensor Model

For fast computations, our inverse sensor model is a piece-wise linear function based on [Loop et al., 2016] and illustrated in Fig. 4.4 (a) operating directly in log-odds space. Similarly to [Loop et al., 2016], we use a model where the measured surface position matches a log-odds value of zero.

We model depth uncertainty σ as a function of measured depth z, assuming it to be linearly or quadratically growing depending on the sensor type (see Fig. 4.4 (b)): quadratic for RGB-D and linear for LiDAR or synthetic (perfect) depth cameras. We relax the assumption of quadratic relation made in [Loop et al., 2016] between the surface thickness $\tau(z) \propto z^2$ with a linear model $\tau(z) \propto z$ bounded by a minimum and maximum surface thickness τ_{min} and τ_{max}, to avoid overgrowing of distant objects.

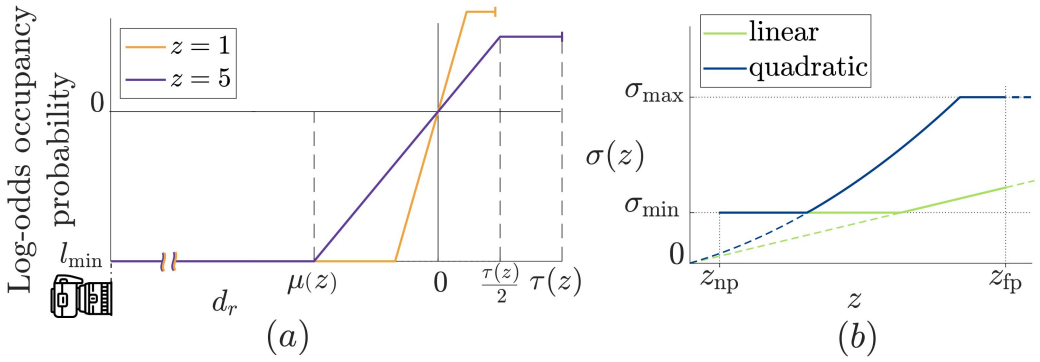

Figure 4.4: (a) Inverse sensor model for two measurements (1m, 5m) expressed as a function of the difference d_r from a query point to the measured surface along the ray. Log-odd values in front of the surface are clipped at l_{min} reached at $\mu = 3\sigma$ and grow linearly up to half the surface thickness $\tau(z)$. (b) Distance-dependent growth of two sensor uncertainty models (linear and quadratic) within the minimum and maximum sensor range z_{np} and z_{fp} and sensor uncertainty σ_{min} and σ_{max}.

4.2.3 Adaptive-Resolution Volume Allocation

The data structure is interpreted as a non-uniform partition of a continuous occupancy 3D scalar field [Vespa et al., 2018]. Consequently we do not assign any

probabilistic meaning to the size of the voxel storing a particular value and instead aim at choosing a partition of the tree which can represent the underlying continuous function up to a certain accuracy.

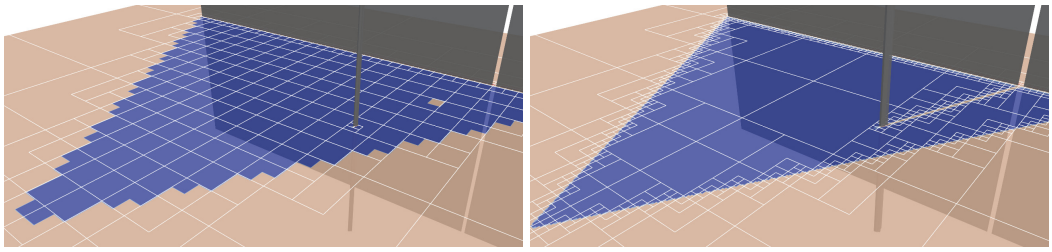

Figure 4.5: Comparison of free space volume allocation in OFusion [Vespa et al., 2018] (**left**) and our new adaptive-resolution strategy (**right**) in an environment with a wall and vertical pole. Blue corresponds to free space and white lines indicate allocated voxels on a map slice. While OFusion ignores occupancy variations inside a voxel's volume, leading to erroneous assumptions of free space behind the pole, our method naturally integrates the volume at the appropriate scale, allocating obstacles and boundaries to unknown space at fine resolution, while representing free space at the coarsest possible level.

Choosing this partition -on the fly- in a volumetric space is not a trivial task. Methods like *OctoMap* [Hornung et al., 2013] use a ray-casting scheme to allocate densely the observed space, simplifying resolution in a later stage called tree pruning. Newer methods like *OFusion* [Vespa et al., 2018] and *UFOMap* [Duberg and Jensfelt, 2020] mitigate the need for dense allocation by using a set of heuristics to choose the needed resolution during the allocation step. The main problem with these approaches is illustrated in Fig. 4.5 for a pinhole camera model. The fact that resolution is chosen as part of the ray-casting and mainly as function of distance tends to ignore changes in occupancy that happen because of depth changes between pixels, which often require high resolution allocation to be accurately captured, leading to parts of space being erroneously labeled as free. This is particularly important in occlusions caused by thin objects and frustum boundaries.

To solve this problem we take a radically different approach, discarding the raycasting and using a so called 'map-to-camera' allocation and updating process, thereby

following an 'as coarse as possible, as fine as required' mapping scheme. Our contribution can be adapted for a pinhole camera model and LiDAR sensor. For better readablity we are describing the overall allocation method for the pinhole camera and describe the changes required for LiDAR sensors afterwards.

Given a new depth image $\mathbf{D}_k$, we start from the root of the octree analysing each node recursively in order to decide whether the variation of occupancy log-odds inside the node meets a bounding criterion:

$$\max |l_k({}_W\mathbf{p}_i) - l_k({}_W\mathbf{p}_j)| < \epsilon \tag{4.7}$$

for every ${}_W\mathbf{p}_i$, ${}_W\mathbf{p}_j$ in the node volume. This criterion, which has been used for 3D data compression on an octree [Knoll et al., 2006], bounds the occupancy error from a sampling perspective, a desirable feature for safe navigation maps. If it is met, we update the node at the given scale; otherwise, it is split into its eight children, and the process is recursively repeated until (4.7) is satisfied or voxel block level is reached.

Evaluating (4.7) naively would require a high resolution sampling of the model in the node's volume, considering all pixel ray measurements that traverse it. However, we observe that, due to the particular structure of the inverse sensor model, we can make a conservative decision based on the measured depth z and the point's query depth $r = \mathbf{T}^{-1}_{WC_k} {}_W\mathbf{p}|_z$ on whether to split the node by only considering the span of measurements $[z_{\min}, z_{\max}]$, and query depths $[r_{\min}, r_{\max}]$ in the node's volume, as illustrated in Fig. 4.6.

While the span $r_{\min,\max}$ can be easily computed from the node's position and size, $z_{\min,\max}$ is more involved. To compute the latter, the node is projected into the depth image defining a bounding box (BB). More importantly, instead of evaluating all pixels within the BB, we sample a pre-computed set of min-max pooling images.

Each pooling image $\mathbf{P}^f_k \in \mathbb{R}^{H \times W}$ with $f \in \{1, \dots, f_{\max}\}$ aggregates the information of

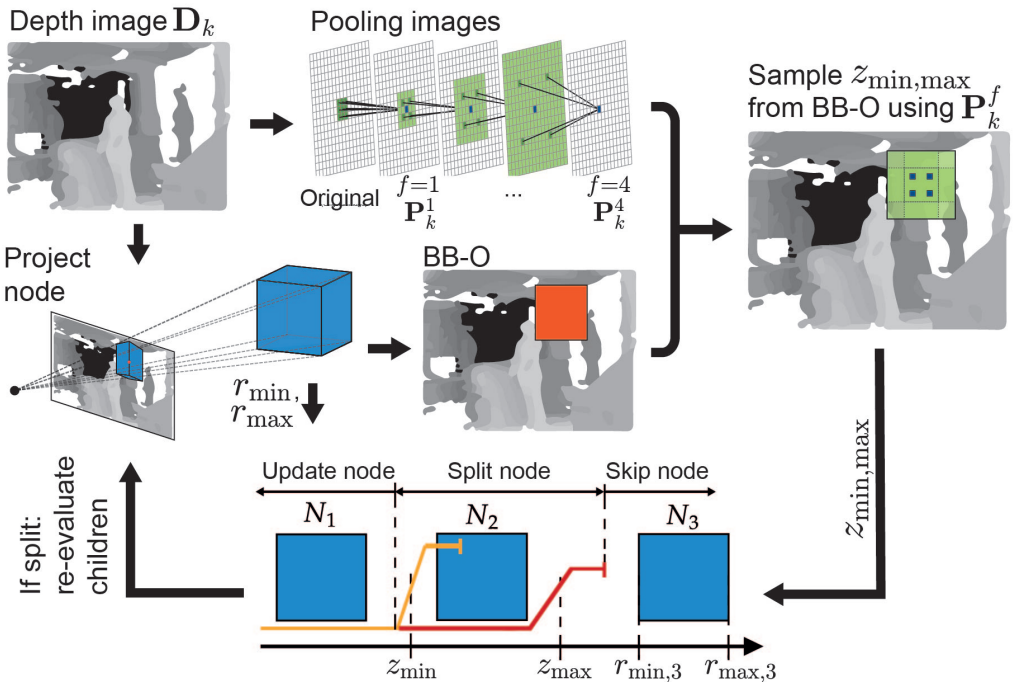

Figure 4.6: The process to decide whether a node should be updated, split or skipped. From the projection bounding box (BB-O), simple queries $\mathbf{P}_k^f[u,v]$ to the pooling image at the correct scale f provide an span of measurements $[z_{\min}, z_{\max}]$ which is used alongside the inverse sensor model and the span of query depths $[r_{\min}, r_{\max}]$ in the node's volume to make a decision (shown for 3 possible cases $N_1 - N_3$). The node is also split if pooling indicates it projects into partly unknown data or crosses the frustum.

the original depth image for different square areas $A^f = (2^f + 1)^2$ centred around each pixel with coordinates u and v. More specifically, pixel $\mathbf{P}_k^f[u,v]$ holds the span of measurement $z_{\min,\max}$ within A^f, as well as a validity and an image crossing state to handle invalid data and image boundaries. These pooling images are computed recursively from the previous level by summarising $\mathbf{P}_k^{f-1}[u \pm 2^{f-2}, v \pm 2^{f-2}]$ as shown in Fig. 4.6. Once this structure is computed, a conservative evaluation of (4.7) can be made by simply computing a BB of the projected node in the image and querying the max-min pooling at the level with the maximum square size that is still fully contained in the BB, reducing the amount of queries needed to decide whether to split the node to no more than 4 in the majority of cases. In summary, by considering measurement spans we reduce the problem of evaluating (4.7) from

a 3D sampling to a 2D one, which is further reduced to a few samples using a pre-computed set of pooling images.

Long range measurements from a LiDAR cover a much larger amount of free space than an RGB-D camera. Due to a larger FoV, it is more likely for LiDAR rays to hit surfaces at shallow angles than those of RGB-D cameras, which results in aliasing artefacts. To adapt the method to LiDAR sensors, a spherical projection image has to be computed and the depth measurements are replaced by the sensor's range measurements. One major difference to the pinhole camera is the 360°horizontal wrap-around of the image. Consequently a projection crossing the vertical image boundaries does not correspond to a frustum crossing. The same property has to be considered during the computation of the pooling image, such that the square BB of the pooled pixels take the wrap around effect into account. To select the integration scale for each ray, we consider the minimum angle between two adjacent LiDAR scan rays, which in turn defines a circular cone. We then determine the scale of the largest voxel volume that fits inside this cone.

4.2.4 Multi-resolution Probabilistic Occupancy Fusion

While data at node level may be at various resolutions, once voxel block level is reached, each block is evaluated at a common resolution scale s_c, further improving performance, reducing aliasing effects and simplifying interpolation required by operations like raycasting and meshing. Similar to [Vespa et al., 2019], measurements are integrated at a mip-mapped scale in the voxel block based on the current distance from the block centre to the camera. Moreover, we integrate blocks that are known to only contain frontiers from free to unknown space not finer than scale s_f. This way, the maximum level of detail ($s_f >= 0$) of the frontier boundaries can be chosen independently of the surface (see Fig. 4.3 for $s_f = 1$).

The desired integration scale s_d may change with the distance. In contrast to [Vespa

et al., 2019], we apply a scale change hysteresis requiring the camera to move a certain amount. Given the focal length c and the map resolution v_{res}, the camera has to move $\Delta_C \mathbf{m}_z = 0.25 \times c \times v_{\text{res}}$ closer to or further away from the voxel block before changing the scale again, to avoid constant scale changes of blocks located at a scale boundary (see Fig. 4.7).

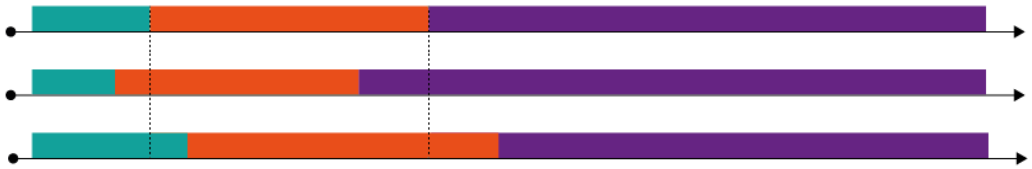

Figure 4.7: The distance based integration scale hysteresis illustrated for the three integration scales 0 (green), 1 (orange), 2 (purple). (top) The initial integration scale boundaries. (middle) The integration scale boundaries when changing to a finer scale. (bottom) The integration scale boundaries when changing to a coarser scale.

We adapt the propagation strategies applied in [Vespa et al., 2019], to our occupancy map representation to keep the hierarchy consistent between scale changes. We replace the parent with the mean of all observed children when up-propagating information to a coarser mip-mapped scale ($s_d > s_c$). However, when down-propagating to a finer integration scale ($s_d < s_c$), we allocate the data first and assign the parent's value to all its children. In either case, we do not change scale immediately. Instead we wait for the block to be fully projected into the image plane multiple times and observed at the desired scale s_d. During this time we update the data at both scales s_c and s_d. Once the changing condition is fulfilled we switch to the new integration scale, deleting the previous buffer. This reduces artefacts by smoothing values during initialisation and preventing blocks that are occluded or just about to exit the camera frustum from changing scale. In comparison to prior work our new down-propagation strategy enables eliminating all helper variables stored in each voxel, thereby drastically reducing memory consumption.

Once all information from a depth image $\mathbf{D}_k$ is integrated into the map, the mean

values of all updated blocks are up-propagated to the mip-mapped scale within each block to enable tri-linear interpolation between neighbours of different scales. Additionally, the maximum occupancy of all updated blocks and nodes is propagated through all levels up to the root of the tree to support fast hierarchical free space queries. Finally, pruning is applied to merge node children whose weighted log-odd occupancy is close enough according to a lower threshold L_{min}. This extra step simplifies the tree when the occupancies converge to similar values. Fig. 4.8 illustrates the difference between *Multires OFusion* and *OFusion* in terms of allocation and voxel updating strategy.

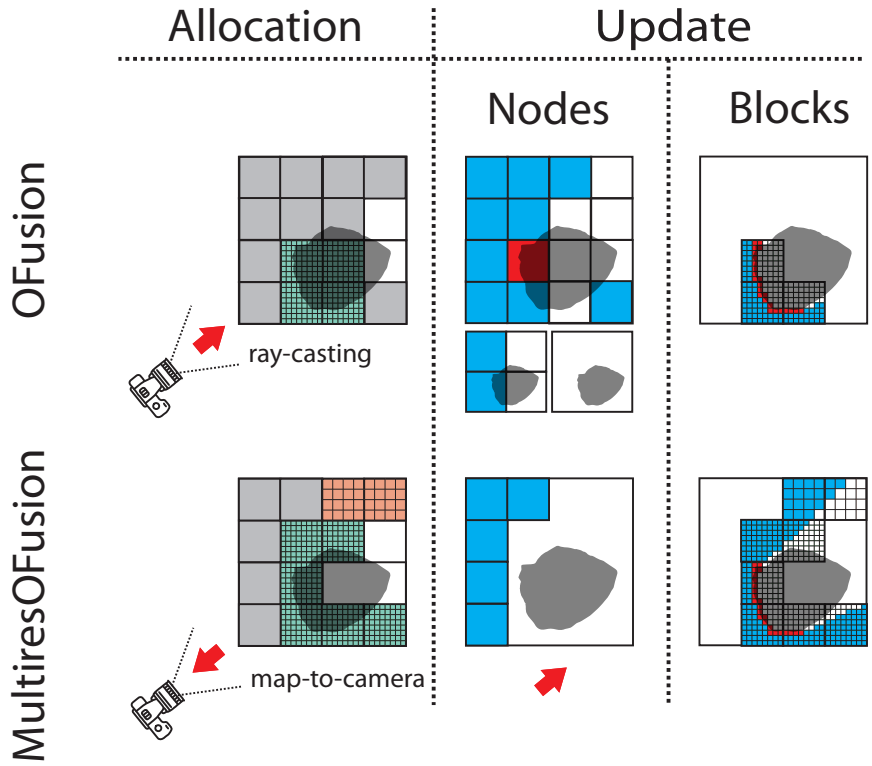

Figure 4.8: The image shows the difference between *Multires OFusion*'s and *OFusion*'s allocation and fusion strategy. While *Multires OFusion* map-to-camera strategy guarantees that the space is densely allocated at the surface as well as boundaries to unknown space, *OFusion* neglects the latter. As a result, *OFusion* allocates the space to unknown areas too coarse, resulting in too optimistic free space approximations. Additionally *OFusion* does integrate data at numerous scales at ones, leading to inconsistency in the occupancy map.

4.2.5 Multi-resolution Ray-casting and Meshing Modules

Fast raycasting is required to render the surface in real-time and/or to enable tracking as part of a full dense SLAM system. With the up-propagated maximum occupancy and observed state stored in our data structure, we implemented a multi-resolution ray-casting strategy to quickly move a ray through large volumes of free space [Knoll et al., 2006].

A meshing module is also provided which computes an adaptive resolution dual mesh following [Wald, 2020]. This approach had been adapted to support our two tier data structure.

4.3 Evaluation

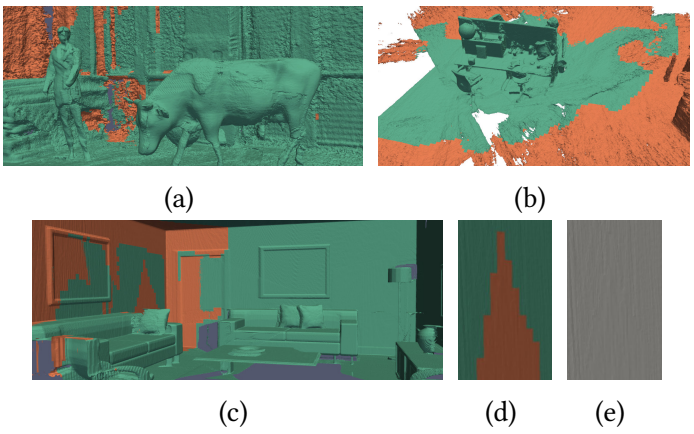

Figure 4.9: Our multi-scale mesh reconstruction for different sensor types with green encoding sampling scale 0 (finest) and orange scale 1. (a) In the *Cow and Lady* dataset and (b) *TUM* dataset, scale is adapted to reflect higher Kinect sensor noise, particularly for places far from sensor trajectory, which loops around the central desk. (c) In the *ICL-NUIM* synthetic dataset a similar model is used to show how, in lack of noise, no artefacts appear due to scale changes. (d) A close up in ICL showing an scale change in a wall. (e) Same as (d) but with uniform color, showing the lack of artefacts in the transition.

This section presents our experimental results. All our tests were performed on an Intel Core i7-8750H CPU operating at 2.20 GHz, 16 GB of memory, and running

Ubuntu 18.04. We used GCC 7.4.0 with OpenMP acceleration for software compilation. For computing the TSDF with *voxblox* [Oleynikova et al., 2017] the *fast* method was used in all comparisons, *OctoMap* [Hornung et al., 2013], *UFOMap* [Duberg and Jensfelt, 2020] and *voxblox* were run in ROS. The parameter values used in the experiments are shown in Table 4.1 unless otherwise specified. Three widely known datasets were used to quantitatively evaluate our system: *TUM* RGB-D [Sturm et al., 2012], *Cow and Lady* [Oleynikova et al., 2017] and *ICL-NUIM* [Handa et al., 2014].

$f_{\max} = 5$ (QVGA) $f_{\max} = 6$ (VGA)	$s_f = 0$ ($v_{\text{res}} = 8$ cm) $s_f = 1$ ($v_{\text{res}} \in \{1 \text{ cm}, 2 \text{ cm}\}$)	
$l_{\text{min,total}} = -100$	$l_{\text{min,iter}} = -5.015$	$L_{\min} = 0.95 \times l_{\text{min,total}}$
$\sigma_{\min} = v_{\text{res}}$	$\sigma_{\max} = 3 \times v_{\text{res}}$	$z_{\text{np}} = 0.4$ m $\quad z_{\text{fp}} = 6$ m
$\tau_{\min} = 3 \times v_{\text{res}}$	$\tau_{\max} = 12 \times v_{\text{res}}$	$w_{\max} = l_{\text{min,total}}/l_{\text{min,iter}}$
Cow and Lady FR3 - Long Office	$\sigma(z) = 0.0025\, z^2$	$\tau(z) = 0.05\, z$
ICL - LR2	$\sigma(z) = 0.05\, z$	$\tau(z) = 0.05\, z$

Table 4.1: Parameter values used in the experiments.

4.3.1 Reconstruction Accuracy

To evaluate how our method performs in terms of surface reconstruction, we evaluate in the *ICL-NUIM Living Room 2* dataset [Handa et al., 2014]. Table 4.2 shows the reconstruction Root Mean Squared Error (RMSE) to the ground truth mesh as computed by the *SurfReg* tool provided by the dataset, with given poses and without extra ICP alignment. Our method outperforms previous approaches on both tested resolutions.

Additionally, we investigate the potential degradation induced by down-sampling the input image. It can be seen the effect on the reconstruction metric in all cases is minor, motivating the use of this later step for improving both running time and memory usage. To show how our method can adaptively select sampling resolution, multiple reconstructions are presented in Fig. 4.9 with color encoding scale.

	Image scale	1 cm	2 cm
Ours	×1	0.0146	**0.0193**
	×0.5	**0.0250**	**0.0341**
SE OFusion	×1	0.0166	0.0259
	×0.5	0.0478	0.0465
SE TSDF	×1	**0.0142**	0.0200
	×0.5	0.0429	0.0367

Table 4.2: Reconstructed mesh RMSE [m] on the *ICL-NUIM LR2* dataset for different resolution and image subsampling.

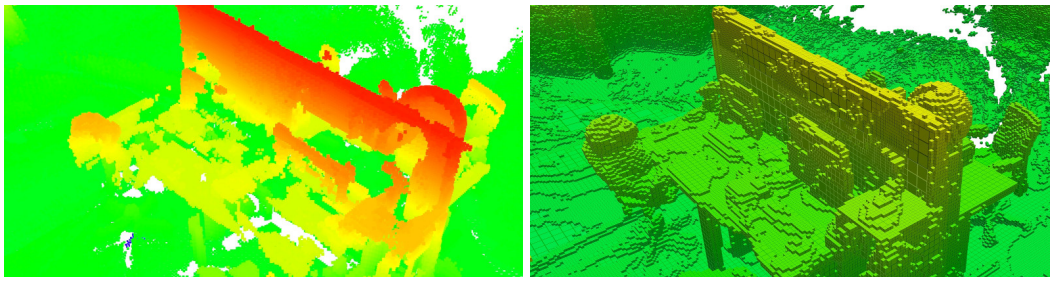

Figure 4.10: Voxelised reconstruction in FR3 - Long Office for *UFOMap* ($n = 0, d \hat{=} 16cm$, their visualiser) (**left**) and our system (**right**). Note the holes in the *UFOMap* surface due to non-conservative free space allocation.

4.3.2 Runtime Performance

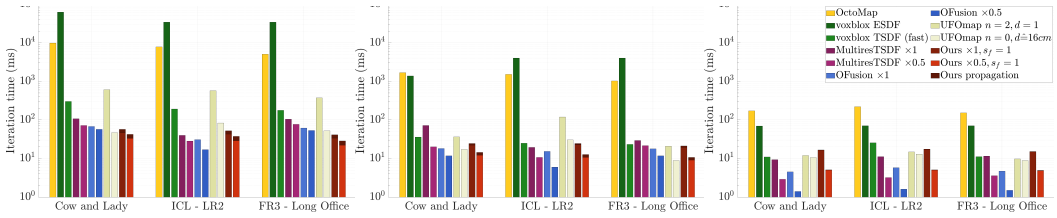

Figure 4.11: Timings for our system alongside other planning libraries, including the widely used *OctoMap* [Hornung et al., 2013], for 1cm (**left**), 2cm (**centre**) and 8cm (**right**) maximum resolution. Notice the logarithmic scale. In the case of *voxblox* both timings for TSDF and ESDF calculation are shown, as both are needed for path planning. For some libraries including ours, results are presented for full resolution input images (×1) and down-sampled images (×0.5). The cost for computing data up-propagation (pooling) in our method is also separated to highlight the cost of having this multi-resolution feature in our map.

Running times for our system and competing methods can be seen in Fig. 4.11. Our method performs best at 1cm resolution for both image configurations, beating

other methods in datasets using real sensors. We attribute this mainly to our adaptive-scale approach which leverages the additional computations needed to accurately and conservatively represent free-unknown space with economising resolution where not needed. As maximum resolution decreases to 8cm, or in small datasets, e.g. ICL-NUIM, the benefits of adaptive-scale are less pronounced while small costs like computing a pooling image become relatively larger. This is expected and consistent with the high-resolution/large-scale environments targeted by the algorithm.

Methods like *OFusion* and *UFOMap* (*fast*) also achieve excellent performance but at the expense of mapping quality, as shown in Section 4.3.5 and Fig. 4.10. Additionally, methods such as *voxblox* present reasonable results, but have the drawback of requiring an extra step of computing an ESDF in order to allow direct planning on them, which becomes prohibitively expensive in high resolution cases. This limits its usability in cases requiring a high resolution navigable map in real-time. Finally *OctoMap* presents significantly higher integration times than newer methods, since it targets low resolution LiDAR made maps.

4.3.3 Multi-resolution, Memory and Efficiency

To assess memory efficiency of our multi-resolution approach, we compare our memory usage to supereight OFusion [Vespa et al., 2018] and supereight MutiresT-SDF [Vespa et al., 2019]. In particular OFusion uses an octree, similarly to our system for storing both free and occupied space, while MultiresTSDF also utilises an octree but for sparsely allocating a narrow-band TSDF.

As a conservative measurement of memory usage, we use the main process' Resident Set Size (RSS) as reported by the Linux kernel to compute the RAM usage.

Results can be seen in Table 4.3. For the intended use case of high resolution reconstruction and planning, our system overcomes competing methods in real life

	Cow & Lady		FR3 Long Office		ICL LR2	
	1cm	8cm	1cm	8cm	1cm	8cm
Ours	**1100**	65	**493**	62	397	105
SE OFusion	1734	**57**	1009	**56**	**192**	**54**
SE MultiresTSDF	2922	65	1728	68	406	56

Table 4.3: Memory consumption in MB at different voxel resolutions.

scenarios, mainly thanks to the adaptive sampling resolution at voxel block level. As resolution decreases, or in small datasets, e.g. *ICL-NUIM*, where the relative voxel block size grows in a way that brings the allocation near to dense, the cost of storing this extra multi-resolution information along with our conservative allocation of frustum boundaries, takes a toll on the usage when compared to simpler methods like OFusion. This extra cost is justified by superior planning performance.

4.3.4 Tracking Performance

In this section, we evaluate our mapping system integrated to a Dense SLAM Pipeline similar to [Newcombe et al., 2011], which is included as an additional component of the presented library.

Table 4.4 shows trajectory accuracy results using the ATE metric against TSDF and OFusion [Vespa et al., 2018] methods on the FR1 - Desk and FR3 - Long Office sequences of the *TUM* RGB-D datasets. The same ICP tracking approach presented in [Newcombe et al., 2011] is used for all pipelines. For point-cloud extraction our system relies on the efficient multi-resolution raycasting method described in Section 4.2.5. Results indicate that maps produced by our approach are suitable for accurate ICP tracking, with the ATE being similar to that of previous methods.

4.3.5 Fast Collision Checking for $\mathbb{R}^3$ Motion Planning

Finally, we evaluate our mapping system as input for path planning. We show that kinodynamically feasible and collision free quadrotor trajectories with map

	ATE (m)			
	FR1 - Desk		FR3 - Long Office	
Pipeline	1 cm	2 cm	1 cm	2 cm
Ours	0.104	0.098	0.194	0.185
SE TSDF	**0.099**	0.103	0.314	FAIL
SE OFusion	0.100	**0.086**	**0.165**	**0.172**

Table 4.4: ATE of various pipelines on *TUM* RGB-D datasets.

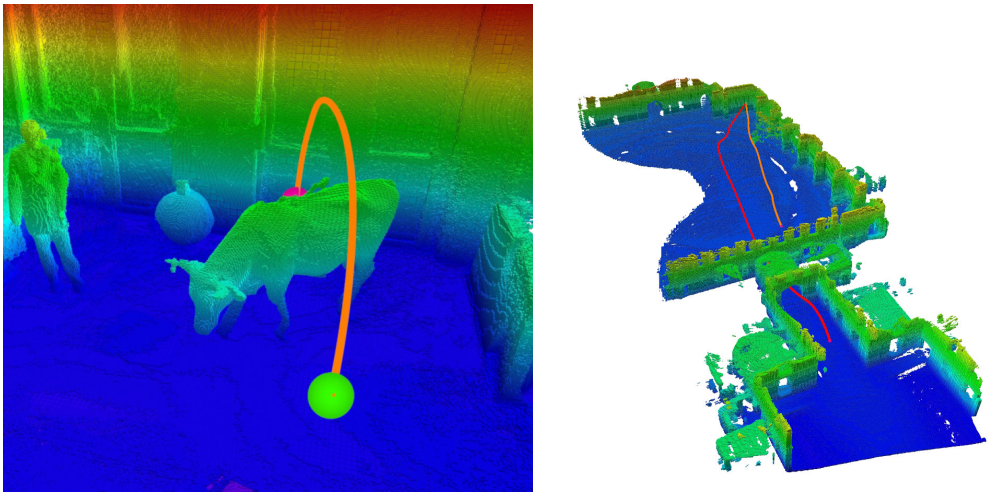

Figure 4.12: (**left**) Planning in the *Cow and Lady* dataset : Trajectory 1 over the cow obtained with our mapping system, OFusion is unable to solve this planning problem due to clutter. (**right**) Planning in The Newer College dataset [Ramezani et al., 2020]: Trajectores 3 (**orange**) and 4 (**red**). Remarkably the method is able to find the shortest path though the centre corridor.

resolutions up to 1cm can be planned in real-time. To guarantee feasibility, we compute a Safe Flight Corridor (SFC) from the start to the end position and optimise 10^{th} order Bernstein polynomial motion primitives within each segment. As a corridor primitive we use cylinders connected by spheres with a minimum radius r_{min}. We use the open motion planning library's (OMPL [Şucan et al., 2012]) informed rapidly-exploring random tree* (informed RRT* [Gammell et al., 2014]) planner to create the SFC connecting the start and end positions. OMPL confirms the safety of each corridor segment by verifying that none of the cylinder and sphere volumes is occupied. This is challenging using a regular volumetric grid where the number of checks grows cubically with the map resolution.

With our multi-resolution maximum occupancy queries, we utilise a 'coarse-to-fine' collision checking approach recursively increasing the resolution in parts of the corridor where needed to reduce the number of checks required.

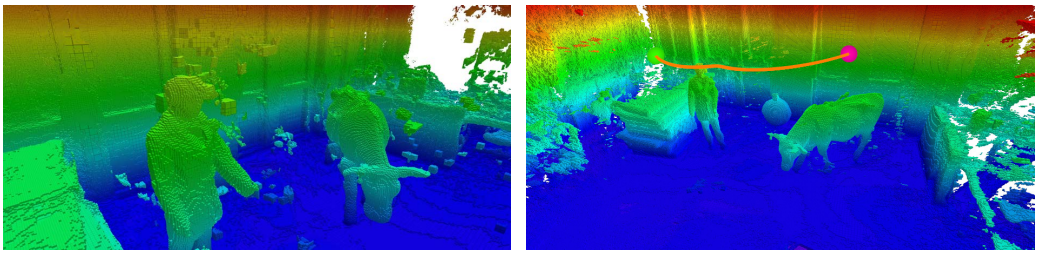

Figure 4.13: (**left**) Clutter in OFusion makes the planner fail in several cases; (**right**) For a different problem (Trajectory 2) both systems find a solution but OFusion struggles with noise while planning times with our pipeline are significantly lower.

We investigate the use of our method for several path planning problems illustrated in Figs. 4.12 and 4.13 and compare against the OFusion library, which also achieves excellent mapping times and provides some multi-resolution output. While our method can find suitable trajectories in every iteration, notably in cluttered datasets like the *Cow and Lady* or large scale like *The Newer College* [Ramezani et al., 2020], OFusion fails in several cases. This is because OFusion lacks a proper mechanism for leveraging information from different scales, instead relying on a simple rule of giving preemptiveness to data in high resolution voxel blocks over values stored at node levels. Given that the latter normally stores information about free space observations and the former allocates surface measurements, OFusion is fast but heavily biased against noise measurements and outliers. In our system we solve this issue by not only propagating data down from nodes to voxel blocks, but also by carefully considering the variation of occupancy inside the node's whole volume (not only the centre value), as described in Section 4.2.3, to avoid having the opposite effect of biasing towards free space.

To further confirm these aspects and highlight the benefits of our library, we modified OFusion to perform dense allocation, i.e. everything integrated at a single

lowest voxel block level. The results in Table 4.5 show the minimum solving time required by the planner to find a SFC. While the dense allocation approach can remove OFusion noise and find suitable trajectories, the planning time required is much higher. This is easily explained by the lack of multi-resolution sampling capabilities, a key feature of our system.

	Traj 1	Traj 2	Traj 3	Traj 4
Ours	**0.01**	**0.1**	**0.06**	**0.4**
SE OFusion	4	FAIL (too noisy)	3	FAIL (too noisy)
SE Dense	5	15	3	6

Table 4.5: Minimum times (sec) to compute 'Safe Flight Corridor'.

4.4 Conclusion

We have introduced a multi-resolution 3D mapping framework that is using an underlying two-tier octree data structure to encode log-odds occupancy probabilities. Thanks to explicit free space encoding in a hierarchical way, the approach supports fast collision checking, crucial in robotic path planning and collision avoidance, while providing high resolution reconstructions simultaneously.

Our framework was evaluated extensively in synthetic and real-world RGB-D datasets: we showed that surface accuracy is competitive with state-of-the-art TSDF-based frameworks, while enabling real-time or near-real-time operation even at centimetre-level resolutions. We finally show our maps used in different real-time 3D trajectory scenarios, including large scale LiDAR ones, to dramatically improve planning time without converting the map into ESDF as required by many existing approaches; thus providing seamless, unprecedented integration between mapping and planning.

In the next chapter we will discuss a more advanced attitude aware trajectory planner, for which we reintroduce the notion of an ESDF. However this ESDF

will be efficiently obtained in an adaptive-resolution manner directly from the multi-resolution occupancy map.

Chapter 5

Orientation Aware Path Initialisation

Contents of Chapter

Parts of this Chapter appear in:

Nils Funk, Juan Tarrio, Sotiris Papatheodorou, Pablo Alcantarilla, Stefan Leutenegger. Orientation Aware Hierarchical, Adaptive-Resolution A* Algorithm for SE(3) UAV Trajectory Planning. *IEEE Robotics and Automation Letters (RA-L), 2023*

5.1 Introduction

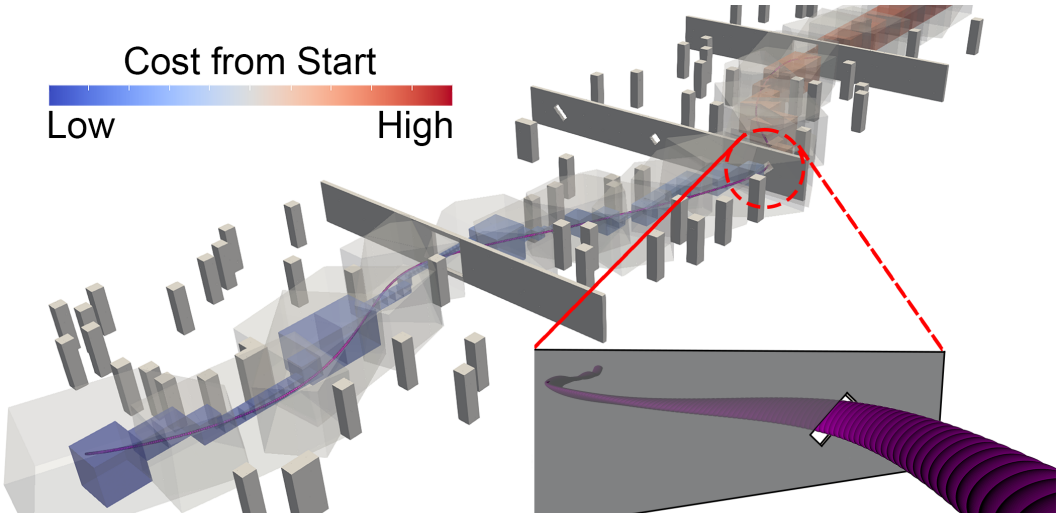

Figure 5.1: Random gap map (map-gap-000) with the adaptive-resolution A* path (with increasing path cost from blue to red), the extracted polyhedron safe flight corridor (transparent white) and the final trajectory (purple). The UAV's attitude has to be taken into consideration during the planning stage to successfully traverse the environment. The zoomed-in area (red) shows the trajectory with attitude (purple) passing through the gap in the wall. The benchmark environment is further described in (Sec. 5.3).

Drone racing challenges like AlphaPilot [Foehn et al., 2022] and IROS Drone Racing Challenge [Moon et al., 2019] expose and help close the gap between human operators and fully autonomous UAV flights. Traditional UAV path planning methods, e.g. [Gao et al., 2018, Zhou et al., 2019], divide the problem into two steps of (1) searching for a feasible path in $\mathbb{R}^3$, i.e. approximating the UAVs shape by a sphere, thus not considering orientation; and (2) trajectory optimisation/refinement considering kinodynamic feasibility. However, for more challenging environments, containing e.g. gaps, the UAV attitude needs to be considered during the first stage for finding a collision-free initial path at all, and therefore the sphere approximation is no longer viable.

Very recently, [Wang et al., 2022] presented a pipeline for orientation aware SE(3) planning which mainly tackles the challenging problem of path optimisation using

a safe flight corridor (SFC) as input. However, significant limitations still remain: a list of waypoints is required for SFC initialisation, which is practical only in limited scenarios where the UAV path is known a priori, e.g. drone racing. Moreover, the SFC search step of the method is slow to compute (see Sec. 5.3 Table 5.2), making it the main bottleneck.

Building up on the mapping strategy of the previous chapter, we introduce a hierarchical top-down A* approach which, to the best of our knowledge, solves for the first time the problem of finding a global minimum cost path in orientation constrained maps, while considering the validity of orientations along the path. The main features of this algorithm are: firstly, the use of an adaptive resolution octree to represent the target cost map and search graph, reducing significantly its complexity in places where it is not required. Secondly, the introduction of a coarse to fine strategy, where a global search is done in coarse resolution, only going to finer resolutions locally where needed. Additionally, in local areas where orientation is critical, we introduce an efficient method to consider the UAV's attitude during the search, based on pre-computed binary masks. Our input cost map can be seamlessly obtained from the *Multi-res Occupancy* representation. By posing this cost as a function of distance to obstacles the planer naturally prefers safer paths, leading to higher success rates.

In summary, we present the following contributions:

1. A fast, orientation aware hierarchical adaptive-resolution A* algorithm capable of finding feasible global paths in challenging environments.

2. A method for very selectively incorporating attitude awareness into A* when needed around narrow gaps.

3. Integration into a working pipeline for SE(3) UAV trajectory planning; hereby achieving significantly faster computation times and higher success rates

compared to the state-of-the-art.

4. Extensive evaluation in randomised environments using a drone racing simulator with a physics engine and closed loop control. See Fig. 5.1 for an example map.

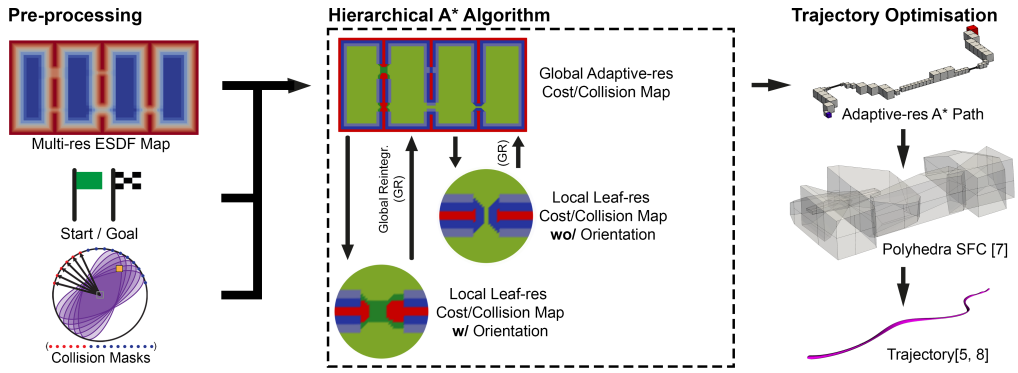

Figure 5.2: Pipeline Overview: In the pre-preprocessing step, the multi-resolution TESDF map (Sec. 5.2.2) and collision masks (Sec. 5.2.4) are computed. All multi-resolution map information is stored in an octree with a finest resolution r_f. Using this input our hierarquical A* method (Sec. 5.2.3) runs on a coarse adaptive resolution grid, going to finer resolution and considering orientation only in local regions where it is needed. The final path is converted into a safe flight corridor (SFC) [Liu et al., 2017b] of polyhedra taking the path's attitude limitations into account. Lastly, the trajectory is optimised as in [Han et al., 2021, Wang et al., 2022].

5.2 Methodology

An overview of our pipeline is illustrated in Fig. 5.2. The overall pipeline can be separated into three stages:

1. *Pre-processing*: A modified version of the hierarchical, multi-resolution occupancy mapping framework from [Funk et al., 2021] is used to generate a multi-resolution truncated ESDF (TESDF) from the occupancy map (Sec. 5.2.2). Equipped with a cost function (Sec. 5.2.3), this field readily translates into a cost map which serves as the input to our cost-based planner. To effi-

ciently account for robot orientation and geometry, UAV collision masks are pre-computed (Sec. 5.2.4). Finally, the start and goal positions are defined.

2. *Path initialisation*: Our novel hierarchical adaptive-resolution A* algorithm searches for a collision free path to the goal position, considering the UAV's attitude if required (Sec. 5.2.5).

3. *Trajectory optimisation*: The free space around the path is approximated using a polyhedra Safe Flight Corridor [Liu et al., 2017b] and the final trajectory is optimised inside it using [Wang et al., 2022, Han et al., 2021] (Sec. 5.2.6).

5.2.1 Robot Geometry

The only restriction we impose on robot geometry is for it to fit inside a solid of revolution. This allows us to ignore the yaw when sampling robot poses.

Along the text, r_{max} refers to the maximum distance from any robot surface point to its centre and respectively r_{min} to the minimum distance. This implies that if the Euclidean distance d from the closest surface to the centre of robot is $d > r_{\text{max}}$ we can be confident the robot is *clear in any orientation*, and oppositely $d < r_{\text{min}}$ implies the robot is in collision *irrespective of its orientation.*

5.2.2 Multi-resolution Cost Map

A multi-resolution 3D cost map is used for the path planning stage. We define the cost at any given point in the map to be a monotonically decreasing function of the distance to the nearest surface, which becomes constant beyond a maximum distance. This allows using a truncated Euclidean signed distance field (TESDF) as the underlying representation. We utilise an octree data structure to store distance information at multiple resolutions for any position $\mathbf{p}$. Starting from a root node at position $\mathbf{x}(\mathbf{p}, s_r)$, covering the entire map, the space is recursively discretised by splitting each node at scale s into eight equally sized children of scale $s - 1$ until

an adaptive leaf scale $s_l(\mathbf{p}) \geq 0$ is reached. The map is initialised using an a priory known multi-resolution occupancy map [Funk et al., 2021].

Following [Oleynikova et al., 2017], we compute the distance values using wave-front propagation. However, different to [Oleynikova et al., 2017], we do not use the quasi-Euclidean distance, but rather keep track of the closest surface node as in [Han et al., 2019]. Occupied nodes are allocated at scale $s = 0$ with a TESDF value d of 0. We use an adaptive leaf scale s_l that increases depending on the surface clearance, sacrificing accuracy for faster computation times. The leaf scale is computed as

$$s_l(d) = \begin{cases} 0, & d \leq r_{\text{max}}, \\ 1 + \frac{d - r_{\text{max}}}{\lambda}, & \text{otherwise.} \end{cases} \tag{5.1}$$

Finally, for the entire map we up-propagate the minimum and maximum distances d_{min} and d_{max} of octree node children along higher scales up to the octree's root. Consequently for any position $\mathbf{p} \in \mathbb{R}^3$ and scale $s \geq s_l(\mathbf{p}) \geq 0$ we can efficiently query min/max distance bounds for all the volume encompassed in the i^{th} node at position $\mathbf{x}_i = \mathbf{x}(\mathbf{p}, s)$. An example TESDF map is illustrated in Fig. 5.3.

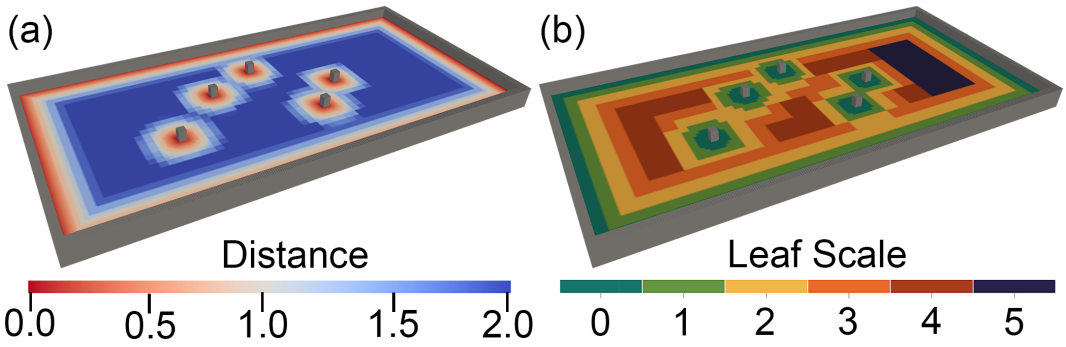

Figure 5.3: Multi-resolution TESDF map of a room with five vertical pillars. (a) A distance slice through the centre of the octree map. The distance is truncated at 2 m. (b) A slice through the centre of the octree map showing leaf scales following Eqn. (5.1) up to the truncation boundary.

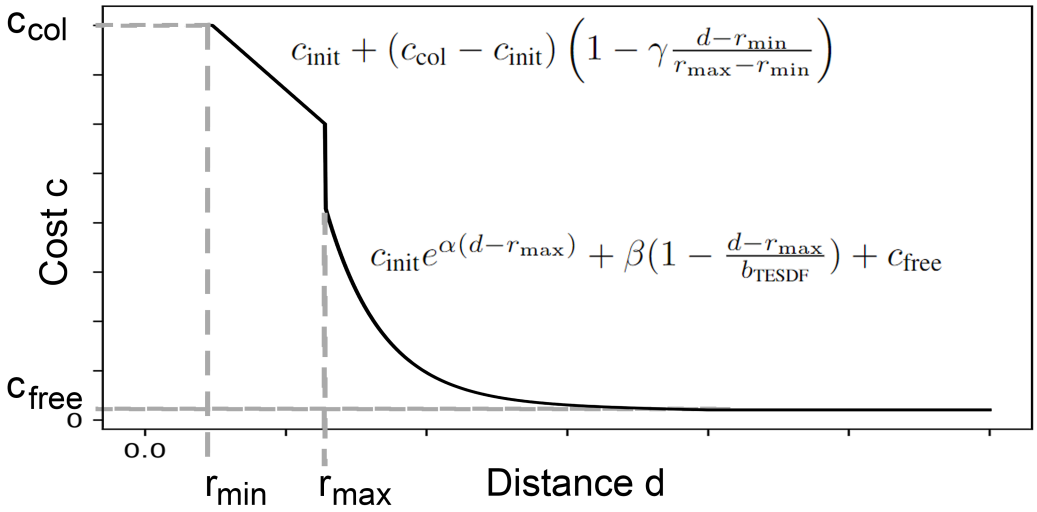

Figure 5.4: The distance-based cost function $c(d)$ used is a combination of a linear part in the region where orientation needs to be considered and a rapidly decaying exponential part afterwards.

5.2.3 Cost Function

In the current formulation any monotonically decreasing function of the distance can be chosen as cost. Naturally however, its shape influences the performance of the algorithm. Our chosen cost function $c(d)$ can be seen in Fig. 5.4. For a surface clearance smaller than the minimum robot radius r_{min} the cost is not defined. For $d \in [r_{min}, r_{max}]$ inclusive, the cost decreases linearly from the maximum collision cost c_{col} until the maximum robot radius r_{max} is reached. Within this interval the robot will be in collision depending on its orientation. Since computing the collision free orientations is expensive and should only be done if required, the high cost of this interval prevents the path to pass through regions that are orientation constrained. From that point onwards the cost drops drastically, decreasing exponentially to a minimum free-space cost c_{free}. Keeping a small $c_{free} > 0$ drives the solution towards minimum path length when far from obstacles.

5.2.4 Attitude Binary Collision Masks

During the path search, we need to repeatedly determine the collision free orientations at certain locations. We therefore discretise the 2D space of possible attitudes – the direction that the symmetry axis faces – as points on the unit sphere which we obtain with Fibonacci lattice sampling (376 samples, illustrated in Fig. 5.5 (a)) yielding a fairly uniform distribution on the sphere.

But instead of computing the collision-free orientations by exhaustively checking all discrete options during path search for a given robot geometry (i.e. when at a surface distance between r_{min} and r_{max}), which would in fact render the algorithm impractically slow, we have conceived the following scheme that forms an integral part of our contribution. The core insight is that given an occupied voxel at a relative offset to the robot, we can pre-compute the (discrete) orientations that are in collision and store them as binary masks, irrespective of the robot's geometry. When checking for orientations in collision at runtime, we can now simply gather all the occupied voxels around the UAV's location to be checked and perform a simple AND operation on the respective cached masks originating from all of the surface voxels. This process is explained in Fig. 5.5.

5.2.5 Hierarchical A*

We aim to find a minimum cost path in 3D space subject to the robot not being in collision at any point along the path. In positions where the collision state depends on the UAV's orientation, we require for at least one discrete orientation to be collision free for the path to be valid. To solve this 3D search problem in feasible time, we resort to two strategies which are described separately: adaptive resolution discretisation of 3D space and coarse to fine optimisation.

In summary, our algorithm starts by running adaptive resolution A* at a coarse resolution, and simultaneously identifying regions that require refinement. For

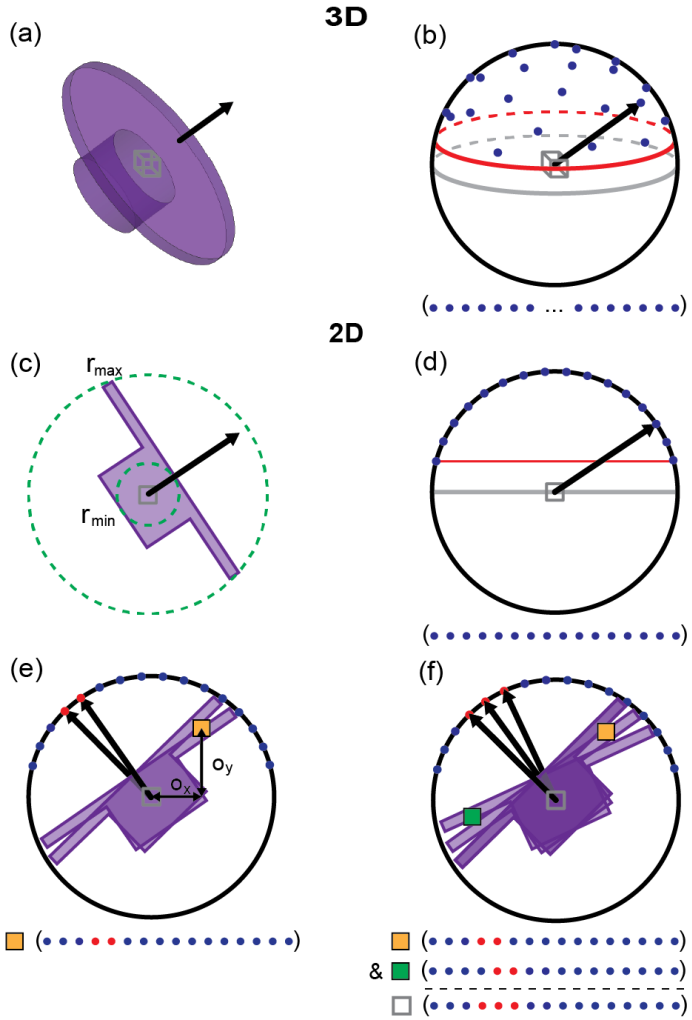

Figure 5.5: Attitude collision masks. (a) Robot approximation by a solid of revolution. (b) Equidistant samples (blue) on a sphere using a Fibonacci lattice. Each sample corresponds to an orientation of the symmetry axis of the robot and is associated with an array index. The plane intersection (red) limits the maximum tilt angle of the robot. (c, d) 2D simplification of (a,b) for illustration. Minimum and maximum robot radii r_{min} and r_{max} (green). (e) Collision mask for example voxel (yellow): attitudes in red cause collision with the voxel (the resulting collision mask is shown below). (f) An example scenario with two surface nodes. The combination of the collision masks fetched based on the offset between surface node and robot.

those smaller regions, defined as bounding boxes, we run a local A* at higher resolutions. If local regions, where orientation needs to be considered, are detected, orientation aware A* is run on those regions only.

A* with adaptive resolution discretisation of space

We formulate the A* search graph following the topology of the cost map octree, where each node is a vertex in the graph. This way, bigger nodes, where cost does not change significantly, are accounted for as a single position, saving significant search time. Given a minimum search scale s_g (with s_g evolving from coarse to fine), we only consider as graph vertices leaf nodes when their scale is $s_l >= s_g$, and else take the nodes at scale s_g instead of finer-scale leave nodes available, effectively clamping the resolution to s_g.

The edges of the graph are based on the octree's 26-neighbour-connectivity. However, due to the adaptive resolution, a node might have fewer or more than 26 neighbours.

To run A* we need to define the transition cost of moving between nodes, i.e. of an edge. Hereby, we want the cost to resemble the line integral of the cost function along the path [Kambhampati and Davis, 1986, Gao et al., 2018]. Specifically, for an edge between two nodes i and j at $\mathbf{x}_i$ and $\mathbf{x}_j$ with respective costs c_i, c_j (computed from the node TESDF values), we define the transition cost:

$$e_{i,j} = \|(\mathbf{x}_i - \mathbf{x}_j)\| \left(r_{i,j}\, c_i + (1 - r_{i,j})\, c_j\right), \tag{5.2}$$

where $r_{i,j} = a_i / \left(a_i + a_j\right)$, with node sizes a_i, a_j.

For nodes which are not leaves the cost is computed based on the maximum up-propagated surface clearance of its children, i.e. this minimum cost inside the node. This is needed to allow the planner to pass through regions that require further refinement, which happens locally when running at finer scales and/or orientation aware mode.

Additionally to the transition cost, we define a heuristic that provides a lower bound

for the cost to goal node $\mathbf{x}_{\text{goal}}$:

$$h_i = \|\mathbf{x}_i - \mathbf{x}_{\text{goal}}\| \, c_{\text{free}}. \tag{5.3}$$

Equipped with these, the A* evaluation function is:

$$f_i = g_i + h_i, \tag{5.4}$$

where g_i is the accumulation of the transition costs $e_{i,j}$ along the path from the start node up to i, computed incrementally as the search progresses.

The A* algorithm operates by exploring the neighbours of the minimal cost node as given by the evaluation function $f(i)$ from a list of nodes to be processed (open list), while keeping track of already processed nodes (closed list). This is done recursively, until the goal node is reached or the open list becomes empty (no solution), for more details see [Hart et al., 1968].

The Node's Collision State

We keep track of additional states in the nodes, in order to implement our coarse-to-fine logic. In particular, we define a collision state function $C(\mathbf{p}, s)$ which summarises the traversability through the node. This function can take 4 possible integer values depending solely on the node's minimum and maximum up-propagated distances ($d_{\text{max}} = d_{\text{min}}$ for leaf nodes), as described next:

$C(\mathbf{p}, s) =$	Name	Condition
1	*NoCol*	$d_{\text{min}} > r_{\text{max}}$
2	*Unk3D*	$d_{\text{max}} > r_{\text{max}} \wedge \; d_{\text{min}} \leq r_{\text{max}}$
3	*Unk5D*	$r_{\text{max}} \geq d_{\text{max}} > r_{\text{min}}$
4	*Col*	$d_{\text{max}} \leq r_{\text{min}}$

While *NoCol* indicates none of the node's children are in collision irrespective of orientation, *Unk3D* indicates at least one child accepts the UAV being there. *Unk5D*

indicates there exists a child where the UAV *may* fit for a reduced set of orientations. *Col* finally indicates the vehicle will crash if at the node. These states are colourised in an example map in Fig. 5.7 (a) for gaps of different width.

Coarse to fine A*

The algorithm starts by running adaptive resolution A* up to a coarse scale s_g. Whilst doing this, for each node we keep track of the maximum collision state of its path predecessors H_j, by propagating it alongside the cost:

$$H_i = \max\left(H_j, C_i\right). \tag{5.5}$$

A finer resolution A* is triggered when during exploration a transition from any *Unk* collision state to *NoCol* happens (i.e. *Unknown* to *No Collision*). In this situation we backtrack the state H_i along the path predecessors back to the last *NoCol* node, defining a local path. We create an inflated bounding box circumscribing the local path, which defines the boundaries of our local A* (bounding boxes are illustrated for different cases in Fig. 5.7 for the example map in Fig. 5.6). The chosen magnitude of the bounding box inflation does not affect the result of our method but influences the overall computational efficiency. Under-inflating the bounding box can create an overhead since multiple neighbouring local A* may run until a large enough area is covered (see Fig. 5.7 (2) where two neighbouring local A*s cover the entire area of the gap). However, over-expanding the bounding box might cause the A* algorithm to traverse the map at high resolution in regions not required.

To set up our local graph at finer resolution, we still make use of pruning to keep the problem simpler. In particular, nodes with known *Col* or *NoCol* states are kept at coarse scale. For nodes with *Unk*, its descendants at fine scale are added instead, in order to disambiguate its collision state. The local open list is initialised with nodes where $H_i = NoCol$ since these nodes have a known global valid collision free path from the start.

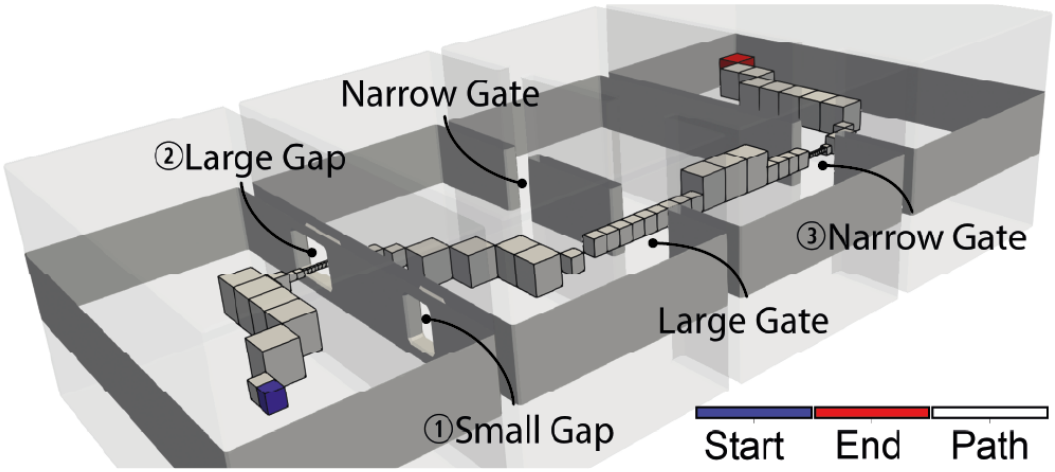

Figure 5.6: Map environment to illustrate the hierarchical A* algorithm. The start and goal are shown in blue and red, respectively, and the final A* path in white. The map contains numerous gaps and gates as indicated. The small gap is too small for the robot to pass through. All other passages can be traversed, but might require the robot's attitude to be taken into account.

Orientation Awareness is triggered only if $H_i = Unk5D$ for the local path. In this case, we fetch the positions of the local surface nodes in the original map and compute the orientation masks in batch for every node with $C_i = Unk5D$, following the methodology from Sec. 5.2.4. If, during a run of a local A* where orientation mask has not been yet computed, a node with $C_i = Unk5D$ is found on a valid path, then the local A* is promoted to *Orientation Aware* by computing the local masks.

Once the local open list is empty, we reintegrate the local changes into the global map to keep the map consistent and optimal. This process is non-trivial. For nodes at the boundary of the bounding box, topology changes have to be accounted for. In particular, finer nodes introduced in the local A* are connected to its neighbours outside the bounding box, generating a new local-global graph. For all boundary nodes whose cost and collision has changed during the local A*, these changes are propagated to all their successor nodes outside in the global map. We add the necessary nodes to the global open list to compensate for the local changes made and continue the global search. Our implementation guarantees that the result

after local A* and propagation is equal to the result that would have been obtained if the new local-global graph topology had been run from the start.

It is important to note that since we use the minimum cost when considering nodes with unknown collision state at coarse resolution, the algorithm will curiously explore such unknown places, only exploring them at higher resolution if they connect two nodes which are known to be traversable. This guarantees that a feasible path will be found if it exists.

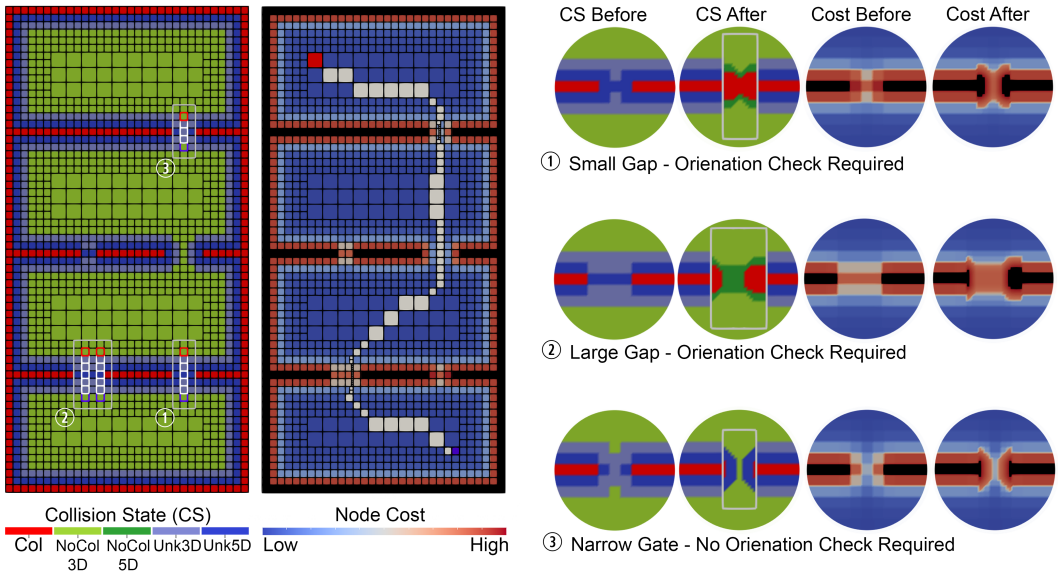

Figure 5.7: Hierarchical A*: (Left) A slice through the map shown in Fig. 5.6 illustrating the collision state and node cost at scale $s \geq s_g$. (Right) Zoom-ins of the three local A* scenarios shown in the global map. When two collision free nodes connect via one or numerous nodes of unknown collision state a local A* algorithm is triggered. An expanded bounding box around the path segment is created to constrain the area of the local A* algorithm. The local A* increases the map resolution to the octree's leaves; and attitude aware collision checks are computed if needed. Bounding boxes (1) and (2) show scenarios that consider the robot's attitude. (1) the A* tries to find a path through the small gap, however after computation of the attitude aware collision states, no path through the gap can be found. (2) Follows the same principle as (1) but a path with constrained attitudes can be found. (3) A local A* is triggered without the attitude computation necessary. For visualisation purposes, we distinguish the *NonCol* collision state into cases with constrained attitude (*NonCol5D*) and non-constrained attitude (*NonCol3D*).

5.2.6 Safe Flight Corridor and Trajectory Optimisation

Once the path has been extracted, we follow the safe flight corridor generation [Liu et al., 2017b] and trajectory optimisation from [Han et al., 2021, Wang et al., 2022]. We modify the approach by incorporating the path's attitude constraint nodes into the segment selection of the polyhedron computation. To better approximate the free space in the constraint areas, we make sure that each constrained path segment is covered by its own polyhedron (see Fig. 5.8 (red) vs. (blue)). Lastly, we follow to optimise a continuous-time polynomial trajectory within the corridor.

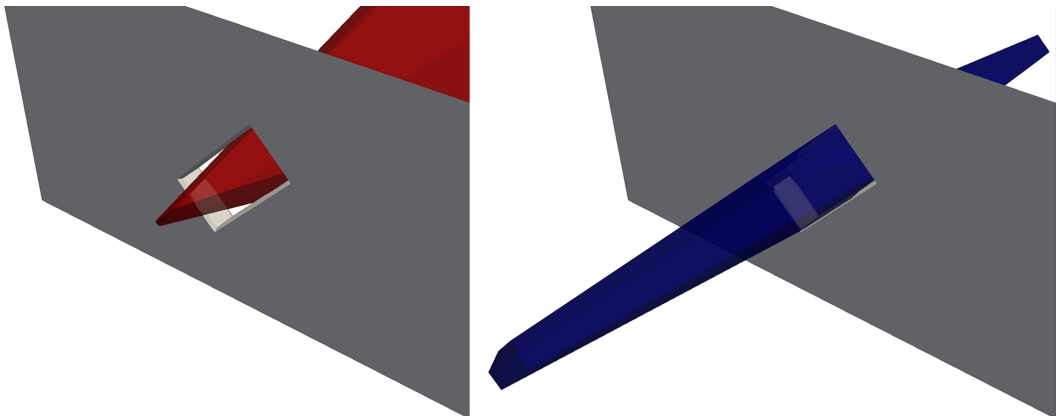

Figure 5.8: Gap approximation with polyhedra: (Red) Our planner not considering the attitude constrains during the polyhedra segment selection. As a result, the free-space inside the gap is not approximated well and causes the MAV to not fit though. (Blue) Ours covering the attitude constrained path segment by its own polyhedron resulting in the most accurate free-space approximation of the gap.

5.3 Evaluation

In this section, we report our quantitative experimental results. To show the robustness of our method in a range of different environments, we have introduced an evaluation protocol based on randomised map generation. This allows us to quantitatively assess the performance in an statistical sense, introducing a new benchmark for the evaluation of this type of method. All experiments use synthetic

environments created in the Unreal Engine Editor [1]. We use three different kinds of racing tracks that can be summarised as follows.

- *Random gate map*: Representing a cluttered environment. The map contains 64 randomly positioned pillars separated by three walls. Each wall has a gate of 4m × 4m positioned randomly at a left, right or centre position. At least one path to the goal exists that does not impose attitude constraints on the UAV to be safely navigated.
- *Random gap map*: Representing a cluttered environment with traversable gaps. The map contains 64 randomly positioned pillars separated by three walls. The first and last wall have a gate of 4m × 4m positioned randomly at a left, right or centre position. The middle wall has three oblique gaps. While all gaps have a length and width larger than the minimum UAV radius r_{min}, only two of the gaps can be traversed. A path to the goal can only by found through a gap, therefore requiring the attitude of the UAV to be considered to be passed safely.
- *Zhangjiajie waypoint navigation* [Han et al., 2021]

We generate ten random gate and gap maps each. All random maps have size of $32 \times 128 \times 4$ m^3 and for both map types only the goal position without waypoints is provided. An example gap map can be seen in Fig. 5.1.

To evaluate our system we use the Drone Racing Simulator presented in [Han et al., 2021]. It combines the popular AirSim [Shah et al., 2018] UAV simulator framework, with a flight controller and map interface. This allows us to evaluate the success rate of our entire pipeline including closed loop control. The test cases are evaluated on a desktop computer with an Intel(R) Core i9-10850K CPU operating at 3.60GHz, 32 GB RAM running Ubuntu 18.04 without any GPU acceleration.

[1] Unreal Engine Editor 4.27: https://www.unrealengine.com/en-US

Throughout the experiments we evaluate our method considering the UAV's attitude (*Ours-Att*) approximating the UAV with an ellipsoid with primary axes r_{max}, r_{max}, r_{min}. We compare against four baselines: a conservative version (*Ours-Cons*) of our hierarchical adaptive-res A* algorithm without considering attitude and approximating the UAV's shape with a sphere of size r_{max}, and a similar optimistic version (*Ours-Opti*) but using r_{min}. We also compare our system to the path initialisation in [Han et al., 2021] with its default values ([Han et al., 2021]-*Def*) and with the obstacles expanded by r_{max} ([Han et al., 2021]-*Cons*). Note that [Han et al., 2021] optimises a shortest path on a regular grid. Throughout the evaluation, the parameter values in Table 5.1 have been used. However, we use the default UAV parameters used in the Zhangjiajie [Han et al., 2021] dataset for the waypoint navigation.

r_{min}	r_{max}	α	β	γ	λ
0.24 m	0.64 m	-5	5	0.5	0.5 m
a_{finest}	s_g	d_b	c_{col}	c_{init}	c_{free}
0.1 m	2	2 m	200	100	5

Table 5.1: Parameter values used in the experiments.

We summarise our core results in Table 5.2. We evaluate the run-time of each approach and the overall trajectory quality, considering trajectory cost (Traj. Cost.), duration (Traj. Dur.) and success rate, defined as the percentage of cases where the UAV is able to fly the provided trajectory successfully under closed loop control. We split the total computation time (Total) into path initialisation (Path Init.), polyhedra generation (Poly. Gen.) and trajectory optimisation (Traj. Opti.). Additionally, we add the time required for the map generation (Map. Gen.) in our cases. However, it should be noted that the time taken to reset and reconnect the search graph for iterative path planning (e.g. way-point navigation) is included in the A* (Path Init.) timings.

In all experiments, our hierarchical adaptive-resolution approaches are approx-

	Map Gen. [ms]	Path Init. [ms]	Poly. Gen. [ms]	Traj. Opti. [ms]	Total [ms]	Traj. Cost.	Traj. Dur. [s]	Success Rate
Gate Map								
Ours-Att	1583	187	**13**	3426	5216	**1372**	**13.63**	**100%**
Ours-Opti	**954**	**125**	**13**	**2465**	**3574**	1861	14.56	**100%**
Ours-Cons	1583	133	**13**	3472	5208	**1372**	**13.63**	**100%**
[Han et al., 2021]-*Def*	-	19133	63	7129	32160	2986	16.69	0%
[Han et al., 2021]-*Cons*	-	24806	72	5195	34053	2160	16.68	60%
Gap Map								
Ours-Att	1590	241	**13**	**2791**	**4660**	**1540**	12.38	**100%**
Ours-Opti	**974**	**132**	**13**	3624	4760	1697	**11.96**	20%
Ours-Cons	-	-	-	-	-	-	-	0%
[Han et al., 2021]-*Def*	-	27377	76	15556	39460	4871	20.70	0%
[Han et al., 2021]-*Cons*	-	-	-	-	-	-	-	0%
Zhangjiajie								
Ours-Att	8913	**187**	**1915**	**20627**	**31642**	**4284**	**34.71**	**100%**
[Han et al., 2021]-*Def*	-	13867	2927	**20514**	37309	**4246**	**36.67**	**100%**

Table 5.2: Evaluation of the test environments

imately *two orders of magnitude faster* than the path search on a regular grid, completely shifting the pipeline bottleneck to the optimisation part. Although global map generation (ESDF) takes a meaningful time too, it should be noted that it could be integrated into the occupancy mapping stage of a full navigation pipeline and done incrementally as in [Oleynikova et al., 2017, Han et al., 2019]. We consider this future work. *Ours-Opti*'s faster map generation is due to the smaller UAV radius $r_{\max}$. The earlier scale change according to Eqn. 5.1 accelerates the wave-front propagation. The path initialisation for our optimistic and conservative approaches are faster than *Ours-Att* as they do not require to compute the attitude dependent collision states online. However, this drastically reduces the trajectory's success rate.

Ours-Att is the only approach taking the UAV shape and attitude into account during the path initialisation, and it achieves a 100% success rate in all test cases. Both conservative pipelines fail for the gap map evaluation as the conservative

UAV shape approximation cannot find a path through the gaps. In contrast, the optimistic approaches often find an initial path through a too narrow gap, resulting in a non feasible trajectory, therefore reducing the overall success rate to 0-20%. Even though the optimistic approaches sporadically find a path through a traversable gap, the trajectory is often in collision. This is mostly due to a poor segment selection during the polyhedra computation. *Ours-Att* takes the attitude constrained path nodes into consideration when selecting the polyhedra segments. Therefore, *Ours-Att* approximates the free space in the constrained areas better. The overall success rate of [Han et al., 2021]-*Def* and [Han et al., 2021]-*Cons* is low, because the shortest path optimisation causes the initial path to pass close to obstacles. As a result, the polyhedra free-space approximation around the path is poor, significantly constraining the trajectory optimisation. While our work focuses on path initialisation, we also note improvements in the trajectory optimisation time: this is caused by better free-space approximation, since the optimisation itself has not been modified. Given that our cost function balances path length and clearance, tight spaces are avoided if not required. Therefore, the safe flight corridor provides a less constraining free space approximation to the optimiser. Given the high number of waypoints in the Zhangjiajie track the path initialisations for both *Ours-Att* and [Han et al., 2021]-*Def* are similar. As a result the trajectory quality and optimisation times are roughly the same for both approaches.

5.4 Conclusion

We have introduced a trajectory planning pipeline that can successfully find optimal global paths in unstructured environments, including narrow, orientation-constraining areas, such as gaps. As a core enabler, we leverage the classical A* method, but formulate it in an adaptive resolution representation of the world, seamlessly integrating with our previous mapping work. Additionally, we use a hierarchical approach, where fine resolution and attitude is only explored where

needed. Finally, our pre-computed collision masks concept allows efficient consideration of attitude during the path search. Paired with safe-flight corridor generation and SE(3) trajectory optimisation, we quantitatively evaluated the approach in simulation – revealing significantly higher success rates than previous approaches thanks to considering orientation from the beginning. Moreover, the path search requires substantially less time to compute, thanks to the extensive use of adaptive and multi-resolution.

Chapter 6

Conclusions

This chapter summarises the main contribution of the thesis. We then provide an overview of the limitations of our work and make suggestions regarding future work.

Contents of Chapter

6.1 Summary of contributions

The contribution of this thesis focus on two domains of UAV navigation: i) the design of a efficient occupancy mapping framework with explicit free-space representation for robot motion planning and ii) a hierarchical attitude aware path initialisation method for UAV SE(3) trajectory optimisation in complex environments. The contributions are summarised in the following:

- In Chapter 4, we introduce our efficient multi-resolution 3D mapping framework for dense probabilistic occupancy mapping and high-resolution accurate surface reconstruction. The underlying two-tier octree data structure encodes the log-odds occupancy probabilities at an adaptive-resolution and differentiates between observed occupied and *free* space. Furthermore, the explicit free-space representation in a hierarchical way allows for multi-resolution occupancy queries for fast online collision checks. Our novel map-to-image allocation algorithm and integration method, seamlessly adapts the map to different scene scales and sensor modalities like pinhole camera and LiDAR sensor models. The integration algorithm selects resolution by constraining the induced sampling error in the observed occupancy consistently, splitting the octree in a coarse-to-fine fashion until the desired accuracy is reached. Using a multi-scale max-min pooling of the input depth image allows real-time operation by providing a conservative indication of measured depth variation in any given volume with only a few queries. We present a comprehensive evaluation with respect to the state-of-the-art approaches on synthetic and real-world RGB-D datasets. Our framework improves runtime performance and memory efficiency without introducing reconstruction artefacts. Furthermore, surface accuracy is competitive with state-of-the-art TSDF-based frameworks, while enabling real-time or near-real-time operation even at centimetre-level resolutions. We finally show our maps used

in different real-time 3D trajectory scenarios, including large scale LiDAR ones, to dramatically improve planning time without converting the map into ESDF as required by many existing approaches; thus providing seamless, unprecedented integration between mapping and $\mathbb{R}^3$ planning.

- In Chapter 5 we introduced a path initialisation method that searches for a global minimum cost path in attitude constrained maps, while considering the validity of orientations along the path. Our orientation aware hierarchical adaptive-resolution A* algorithm enables a top-down approach, selectively incorporating fine resolution and attitude awareness where needed. By posing this cost as a function of distance to obstacles the planner naturally prefers safer paths. The underlying cost field can be seamlessly obtained from the *Multi-res Occupancy* framework presented in Chapter 4. Finally, our pre-computed collision masks concept allows efficient consideration of attitude during the path search. Paired with safe-flight corridor generation and SE(3) trajectory optimisation, we integrate our approach into a working pipeline for SE(3) UAV navigation. We extensively evaluate our approach in randomised environments using a drone racing simulator with a physics engine and closed loop control; hereby achieving significantly faster computation times and higher success rates compared to state-of-the-art approaches thanks to the extensive use of adaptive and multi-resolution and consideration of orientation during the path initialisation.

6.2 Limitations & Future Work

The following section provides a summary of the limitations and provides suggestions for future work.

Deep Learning Depth Image & Map Completion

Unfortunately the incomplete depth images of the RBD-D camera cause numerous

difficulties. Firstly, the efficiency of our *map-to-camera* allocation strategy is heavily dependent on the density of the depth image, as a sparse depth image renders the integration slow, due to the fine allocation around all pixel projection with unknown neighbours. Furthermore, gaps of unknown space in the resulting map will limit the motion planning. In future work this problem can be tackled by prior depth image completion as suggested in [Popović et al., 2021]. Alternatively, the incorporation of scene completion methods provide a solution to the latter challenge.

Incremental ESDF Generation

As mentioned in Chapter 5 our adaptive-resolution cost map is generated from pre-mapped environments and does not support incremental updates at the current time. The missing mechanism to incrementally update the ESDF map has been solved in the past for single resolution maps [Han et al., 2019] [Oleynikova et al., 2017]. We therefore consider the incremental ESDF update using our data structure to be mainly an engineering implementation problem.

Kinodynamic Feasibility of Path Initialisation

While our attitude aware A* algorithm greatly increases the success rate compared to the conventional path initialisations, it does not guarantee that a kinodynamically feasible trajectory can be found without violating the polyhedron constraints considering the UAV's kinodynamic limitations. Therefore, it should be noted that it may occur that the trajectory optimisation will fail to find a collision-free kinodynamically feasible final trajectory. While this challenge is non-trivial to solve, we think a feedback mechanism incorporating the critical points of trajectory optimisation into the cost map, provides a sensible way to find other topological path initialisations to be evaluated in case of collision.

GPU Acceleration

Our algorithms mainly require sequential execution (A* algorithm, ESDF wave-front propagation) and are therefore not suitable to run on a GPU. However a GPU

accelerated implementation of our highly parallisable mapping framework might allow further improvements in terms of computation speeds.

On-board State Estimation

Even though the state estimation aspect is not the focus of our thesis it is an integral part of the success of the navigation. The ICP tracking algorithm used in our implementation is not suitable for our application for high speed camera movements. Therefore updating the state estimation with a state-of-the-art visual inertial framework suitable for aggressive flight maneuvers should greatly improve the overall navigation framework to work in real-world environments.

Dynamic Environments

At the current time our motion planning framework does not consider dynamic objects. Consequently the global trajectories does assume a static environment for the entire planning horizon. UAV trajectory optimisation is a new but active research topic in terms of representation and optimisation methods. We are convinced that our underlying probabilistic occupancy map representation provides a promising map representation in regards to motion planning in dynamic environments.

Exploration

Central to our path initialisation method is a fully observed environment. Other than the incremental cost update, the integration of unknown space in our path initialisation and SFC generation method is required to navigate real-world-environments. Especially considering a "safe" planner which threats unknown space as occupied, a clever exploration algorithm will be required to expand the planning horizon and replanning strategy to reach the goal position.

Bibliography

[Behnke, 2004] Behnke, S. (2004). Local multiresolution path planning. In Polani, D., Browning, B., Bonarini, A., and Yoshida, K., editors, *RoboCup 2003: Robot Soccer World Cup VII*, pages 332–343, Berlin, Heidelberg. Springer Berlin Heidelberg.

[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 *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1476–1483.

[Chen et al., 2022] Chen, Y., Lai, S., Cui, J., Wang, B., and Chen, B. M. (2022). Gpu-accelerated incremental euclidean distance transform for online motion planning of mobile robots. *IEEE Robotics and Automation Letters*, 7(3):6894–6901.

[Cowlagi and Tsiotras, 2012] Cowlagi, R. V. and Tsiotras, P. (2012). Multiresolution motion planning for autonomous agents via wavelet-based cell decompositions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(5):1455–1469.

[Deits and Tedrake, 2015] Deits, R. and Tedrake, R. (2015). Efficient mixed-integer planning for uavs in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49.

[Ding et al., 2019] Ding, W., Gao, W., Wang, K., and Shen, S. (2019). An efficient b-spline-based kinodynamic replanning framework for quadrotors. *IEEE Transactions on Robotics*, 35(6):1287–1306.

[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.

[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.

[Elfes, 1989] Elfes, A. (1989). *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* PhD thesis, Carnegie Mellon University.

[Falanga et al., 2017] Falanga, D., Mueggler, E., Faessler, M., and Scaramuzza, D. (2017). Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5774–5781.

[Felzenszwalb and Huttenlocher, 2012] Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). Distance transforms of sampled functions. *Theory of Computing*, 8(19):415–428.

[Foehn et al., 2022] Foehn, P., Brescianini, D., Kaufmann, E., Cieslewski, T., Gehrig, M., Muglikar, M., and Scaramuzza, D. (2022). Alphapilot: Autonomous drone racing. *Autonomous Robots*, 46(1):307–320.

[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.

[Gammell et al., 2014] Gammell, J., Srinivasa, S., and Barfoot, T. (2014). Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic. *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*.

[Gao et al., 2017] Gao, F., Lin, Y., and Shen, S. (2017). Gradient-based online safe trajectory generation for quadrotor flight in complex environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3681–3688.

[Gao and Shen, 2016] Gao, F. and Shen, S. (2016). Online quadrotor trajectory generation and autonomous navigation on point clouds. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 139–146.

[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.

[Gao et al., 2018] Gao, F., Wu, W., Lin, Y., and Shen, S. (2018). Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 344–351.

[Guo et al., 2022] Guo, J., Xun, Z., Geng, S., Lin, Y., Xu, C., and Gao, F. (2022). Dynamic free-space roadmap for safe quadrotor motion planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10523–10528.

[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 *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 4423–4430.

[Han et al., 2021] Han, Z., Wang, Z., Pan, N., Lin, Y., Xu, C., and Gao, F. (2021). Fast-racing: An open-source strong baseline for SE(3) planning in autonomous drone racing. *IEEE Robotics and Automation Letters*, 6(4):8631–8638.

[Handa et al., 2014] Handa, A., Whelan, T., McDonald, J., and Davison, A. J. (2014). A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531.

[Hanover et al., 2023] Hanover, D., Loquercio, A., Bauersfeld, L., Romero, A., Pěnička, R., Song, Y., Cioffi, G., Kaufmann, E., and Scaramuzza, D. (2023). Autonomous drone racing: A survey.

[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.

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

[Hauer and Tsiotras, 2016] Hauer, F. and Tsiotras, P. (2016). Reduced complexity multi-scale path-planning on probabilistic maps. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 83–88.

[Hirata and Kumon, 2014] Hirata, T. and Kumon, M. (2014). Optimal path planning method with attitude constraints for quadrotor helicopters. In *Proceedings of the 2014 International Conference on Advanced Mechatronic Systems*, pages 377–381.

[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(3):189–206.

[Jiang et al., 2023] Jiang, C., Zhang, H., Liu, P., Yu, Z., Cheng, H., Zhou, B., and Shen, S. (2023). H_2-mapping: Real-time dense mapping using hierarchical hybrid representation. *IEEE Robotics and Automation Letters*, 8(10):6787–6794.

[Kambhampati and Davis, 1986] Kambhampati, S. and Davis, L. (1986). Multiresolution path planning for mobile robots. *IEEE Journal on Robotics and Automation*, 2(3):135–145.

[Karaman and Frazzoli, 2010] Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2).

[Karaman and Frazzoli, 2011] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186.

[Kaufmann et al., 2018] Kaufmann, E., Gehrig, M., Foehn, P., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D. (2018). Beauty and the beast: Optimal methods meet learning for drone racing.

[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)*, volume 4, page 1.

[Knoll et al., 2006] Knoll, A., Wald, I., Parker, S., and Hansen, C. (2006). Interactive isosurface ray tracing of large octree volumes. In *IEEE Symposium on Interactive Ray Tracing*, pages 115–124.

[Kong et al., 2021] Kong, F., Xu, W., Cai, Y., and Zhang, F. (2021). Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots. *IEEE Robotics and Automation Letters*, 6(4):7869–7876.

[Larsson et al., 2021] Larsson, D. T., Maity, D., and Tsiotras, P. (2021). Information-theoretic abstractions for planning in agents with computational constraints. *IEEE Robotics and Automation Letters*, 6(4):7651–7658.

[Lau et al., 2010] Lau, B., Sprunk, C., and Burgard, W. (2010). Improved updating of euclidean distance maps and voronoi diagrams. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 281–286.

[LaValle, 1998] LaValle, S. M. (1998). Rapidly-exploring random trees : a new tool for path planning. *The annual research report.*

[Lee and Yu, 2009] Lee, J.-Y. and Yu, W. (2009). A coarse-to-fine approach for fast path finding for mobile robots. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 5414–5419.

[Lin et al., 2019] Lin, J., Wang, L., Gao, F., Shen, S., and Zhang, F. (2019). Flying through a narrow gap using neural network: an end-to-end planning and control approach. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3526–3533.

[Liu et al., 2017a] Liu, S., Atanasov, N., Mohta, K., and Kumar, V. (2017a). Search-based motion planning for quadrotors using linear quadratic minimum time control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2872–2879.

[Liu et al., 2018] Liu, S., Mohta, K., Atanasov, N., and Kumar, V. (2018). Search-based motion planning for aggressive flight in se(3). *IEEE Robotics and Automation Letters*, 3(3):2439–2446.

[Liu et al., 2017b] Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C. J., and Kumar, V. (2017b). Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695.

[Loianno et al., 2017] Loianno, G., Brunner, C., McGrath, G., and Kumar, V. (2017). Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters*, 2(2):404–411.

[Loop et al., 2016] Loop, C., Cai, Q., Orts-Escolano, S., and Chou, P. A. (2016). A closed-form bayesian fusion equation using occupancy probabilities. In *Fourth International Conference on 3D Vision (3DV)*, pages 380–388.

[Lopez and How, 2017a] Lopez, B. T. and How, J. P. (2017a). Aggressive 3-d collision avoidance for high-speed navigation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5759–5765.

[Lopez and How, 2017b] Lopez, B. T. and How, J. P. (2017b). Aggressive collision avoidance with limited field-of-view sensing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1358–1365.

[Mellinger and Kumar, 2011] Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525.

[Mescheder et al., 2019] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4455–4465.

[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 *ECCV*.

[Moon et al., 2019] Moon, H., Martinez-Carranza, J., Cieslewski, T., Faessler, M., Falanga, D., Simovic, A., Scaramuzza, D., Li, S., Ozo, M., De Wagter, C., et al. (2019). Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12(2):137–148.

[Museth et al., 2013] Museth, K., Lait, J., Johanson, J., Budsberg, J., Henderson, R., Alden, M., Cucka, P., Hill, D., and Pearce, A. (2013). Openvdb: an open-source data structure and toolkit for high-resolution volumes.

[Natarajan et al., 2021] Natarajan, R., Choset, H., and Likhachev, M. (2021). Interleaving graph search and trajectory optimization for aggressive quadrotor flight. *IEEE Robotics and Automation Letters*, 6(3):5357–5364.

[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 *IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136.

[Nex and Remondino, 2014] Nex, F. and Remondino, F. (2014). UAV for 3D mapping applications: A review. *Applied Geomatics*, 6(1):1–15.

[Nieuwenhuisen and Behnke, 2019] Nieuwenhuisen, M. and Behnke, S. (2019). Search-based 3d planning and trajectory optimization for safe micro aerial vehicle flight under sensor visibility constraints. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9123–9129.

[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*, 32(6).

[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 *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339.

[Oleynikova et al., 2016b] Oleynikova, H., Taylor, Z., Fehr, M., Nieto, J., and Siegwart, R. (2016b). Voxblox: Building 3d signed distance fields for planning.

[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 *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373.

[Oleynikova et al., 2018] Oleynikova, H., Taylor, Z., Siegwart, R., and Nieto, J. (2018). Sparse 3d topological graphs for micro-aerial vehicle planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9.

[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 *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5331–5338.

[Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174.

[Penicka and Scaramuzza, 2022] Penicka, R. and Scaramuzza, D. (2022). Minimum-time quadrotor waypoint flight in cluttered environments. *IEEE Robotics and Automation Letters*, 7(2):5719–5726.

[Penicka et al., 2022] Penicka, R., Song, Y., Kaufmann, E., and Scaramuzza, D. (2022). Learning minimum-time flight in cluttered environments. *IEEE Robotics and Automation Letters*, 7(3):7209–7216.

[Popović et al., 2021] Popović, M., Thomas, F., Papatheodorou, S., Funk, N., Vidal-Calleja, T., and Leutenegger, S. (2021). Volumetric occupancy mapping with probabilistic depth completion for robotic navigation. *IEEE Robotics and Automation Letters*, 6(3):5072–5079.

[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. *arXiv:2003.05691*.

[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 *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6332–6339.

[Richter et al., 2016] Richter, C., Bry, A., and Roy, N. (2016). *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*, pages 649–666. Springer International Publishing, Cham.

[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.

[Shah et al., 2018] Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2018). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In Hutter, M. and Siegwart, R., editors, *Field and Service Robotics*, pages 621–635, Cham. Springer International Publishing.

[Song et al., 2022] Song, Y., Shi, K., Penicka, R., and Scaramuzza, D. (2022). Learning perception-aware agile flight in cluttered environments.

[Song et al., 2021] Song, Y., Steinweg, M., Kaufmann, E., and Scaramuzza, D. (2021). Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 1205–1212.

[Steinbrücker et al., 2014] Steinbrücker, F., Sturm, J., and Cremers, D. (2014). Volumetric 3D mapping in real-time on a CPU. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2021–2028.

[Stent et al., 2015] Stent, S., Gherardi, R., Stenger, B., and Cipolla, R. (2015). Detecting change for multi-view, long-term surface inspection. In *BMVC*, pages 127–1.

[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 *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 573–580.

[Şucan et al., 2012] Şucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82. http://ompl.kavrakilab.org.

[Thrun, 2003] Thrun, S. (2003). Learning Occupancy Grid Maps with Forward Sensor Models. *Autonomous Robots*, 15(2):111–127.

[Tordesillas and How, 2022] Tordesillas, J. and How, J. P. (2022). Minvo basis: Finding simplexes with minimum volume enclosing polynomial curves. *Computer-Aided Design*, 151:103341.

[Tordesillas et al., 2022] Tordesillas, J., Lopez, B. T., Everett, M., and How, J. P. (2022). Faster: Fast and safe trajectory planner for navigation in unknown environments. *IEEE Transactions on Robotics*, 38(2):922–938.

[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 *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 215–222.

[Vespa et al., 2019] Vespa, E., Funk, N., Kelly, P. H. J., and Leutenegger, S. (2019). Adaptive-Resolution Octree-Based Volumetric SLAM. In *International Conference on 3D Vision*, pages 654–662.

[Vespa et al., 2018] Vespa, E., Nikolov, N., Grimm, M., Nardi, L., Kelly, P. H. J., and Leutenegger, S. (2018). Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151.

[Vizzo et al., 2022] Vizzo, I., Guadagnino, T., Behley, J., and Stachniss, C. (2022). Vdbfusion: Flexible and efficient tsdf integration of range sensor data. *Sensors*, 22:1296.

[Wald, 2020] Wald, I. (2020). A simple, general, and gpu friendly method for computing dual mesh and iso-surfaces of adaptive mesh refinement (amr) data.

[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 *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5041.

[Wang et al., 2021b] Wang, Y., Ji, J., Wang, Q., Xu, C., and Gao, F. (2021b). Autonomous flights in dynamic environments with onboard vision. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1966–1973.

[Wang et al., 2022] Wang, Z., Zhou, X., Xu, C., and Gao, F. (2022). Geometrically constrained trajectory optimization for multicopters. *IEEE Transactions on Robotics*, 38(5):3259–3278.

[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 *2013 IEEE International Conference on Robotics and Automation*, pages 5054–5061.

[Whelan et al., 2012] Whelan, T., McDonald, J., Kaess, M., Fallon, M., Johannsson, H., and Leonard, J. J. (2012). Kintinuous: Spatially Extended KinectFusion. In *Proceedings of Robotics: Science and Systems (RSS)*.

[Ye et al., 2022] Ye, H., Pan, N., Wang, Q., Xu, C., and Gao, F. (2022). Efficient sampling-based multirotors kinodynamic planning with fast regional optimization and post refining. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3356–3363.

[Ye et al., 2021a] Ye, H., Xu, C., and Gao, F. (2021a). Std-trees: Spatio-temporal deformable trees for multirotors kinodynamic planning. *CoRR*, abs/2109.07741.

[Ye et al., 2021b] Ye, H., Zhou, X., Wang, Z., Xu, C., Chu, J., and Gao, F. (2021b). Tgk-planner: An efficient topology guided kinodynamic planner for autonomous quadrotors. *IEEE Robotics and Automation Letters*, 6(2):494–501.

[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 *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4063–4069.

[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.

[Zhang et al., 2019] Zhang, J., Hu, C., Chadha, R. G., and Singh, S. (2019). Maximum likelihood path planning for fast aerial maneuvers and collision avoidance. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2805–2812.

[Zhang et al., 2020] Zhang, J., Hu, C., Chadha, R. G., and Singh, S. (2020). Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, 37(8):1300–1313.

[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.

[Zhou et al., 2020] Zhou, B., Gao, F., Pan, J., and Shen, S. (2020). Robust real-time uav replanning using guided gradient-based optimization and topological paths. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1208–1214.

[Zhou et al., 2019] Zhou, B., Gao, F., Wang, L., Liu, C., and Shen, S. (2019). Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536.

[Zhou et al., 2021a] Zhou, B., Pan, J., Gao, F., and Shen, S. (2021a). Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, 37(6):1992–2009.

[Zhou et al., 2021b] Zhou, X., Wang, Z., Ye, H., Xu, C., and Gao, F. (2021b). Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485.

[Zienkiewicz et al., 2016] Zienkiewicz, J., Tsiotsios, A., Davison, A., and Leutenegger, S. (2016). Monocular, Real-Time Surface Reconstruction Using Dynamic Level of Detail. In *International Conference on 3D Vision*, pages 37–46.

[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.