Communication-Efficient Active Reconstruction using Self-Organizing Gaussian Mixture Models

Kshitij Goel

CMU-RI-TR-24-76

December 2024

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee

Wennie Tabib, *Chair*
David Wettergreen
Sebastian Scherer
Nathan Michael (Shield AI)
Kostas Alexis (NTNU, Norway)

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Keywords: Reconstruction, RGB-D Perception, Distributed Mapping, Multi-Robot Systems, Collision Avoidance, Gaussian mixture models

Abstract

For the multi-robot active reconstruction task, this thesis proposes using Gaussian mixture models (GMMs) as the map representation that enables multiple downstream tasks: high-fidelity static scene reconstruction, communication-efficient map sharing, and safe informative planning. A new method called Self-Organizing Gaussian mixture modeling (SOGMM) is proposed that estimates the model complexity (i.e., number of Gaussian components) based on the depth and intensity data observed by the robots. This method provides higher fidelity maps and lower map sizes compared to prior GMM-based and grid-based methods. Due to the lower map sizes, sharing the maps amongst robots or to a base station is tractable on low-bandwidth communication channels. To enable safe informative planning, a motion primitives-based planner is developed with two robot-robot and robot-environment collision avoidance approaches. Both approaches provide collision avoidance capability without maintaining a global occupancy grid, which can often consume too much memory onboard the robots. Informative planning is enabled via an information-theoretic action selection strategy with the robots sharing plans and map fragments over a serial connection. The final active reconstruction system is demonstrated in a real cave with two quadcopters equipped with RGB-D cameras. The overall approach provides up to two orders of magnitude reduction in communication bandwidth usage while providing higher depth and intensity reconstruction accuracy compared to dense grid-based methods.

Acknowledgements

The following individuals and organizations are acknowledged for their contributions to this thesis.

- *Funding*: Uber Presidential Fellowship by Carnegie Mellon University.
- *Advising and Mentoring*: Wennie Tabib, Nathan Michael, David Wettergreen, Sebastian Scherer, and Kostas Alexis.
- *Research Paper Feedback*: Micah Corah, John Yao, Cormac O'Meadhra, Tabitha Lee, Adiya Dhawale, Arjav Desai, Ellen Cappo, Matt Collins, Margaret Hansen, and Shivam Vats.
- *Software*: Xuning Yang, Alex Spitzer, Jonathan Lee, Vishnu Desaraju, Mosam Dabhi, and Shobhit Srivastava.
- *Hardware*: Curtis Boirum and Yves Georgy Daoud.
- *Cave experiments*: Brian Osbun, Carroll Bassett, Hope Brooks, Ryan Maurer, David Cale, and Dennis Melko.
- *Friends and Family*: Anil Kumar Goel, Shailja Goel, Geetika Goel, Shobha Suryanarayan, Deepa Suryanarayan, Bharat Agarwal, and Juhi Dewan.

This thesis is dedicated to my father, Anil Kumar Goel.

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

This thesis considers the problem of reconstructing an environment using multiple robots, when each robot is equipped with a sensor that can provide dense imaging data (e.g., RGB-D, thermal, depth). This problem arises in applications such as remote search and rescue [87] and planetary exploration [168].

Dense point cloud data generated through depth sensors needs to be mathematically modeled in a way that all the robots in the team can reconstruct the environment efficiently and accurately. For efficiency, the model built by individual robots should be succint enough to pass on limited bandwidth communication channels and store in the memory onboard the robot. For accuracy, the model should possess enough information to enable informative planning and yield high-fidelity environment maps.

Towards achieving these goals, in this thesis we hypothesize that:

> Using Gaussian mixture models as the map representation yields high-fidelity maps, communication-efficient map sharing, and safe informative planning via motion primitives during active reconstruction with multiple robots.

To feasibly evaluate this hypothesis, we make a set of simplifying assumptions (Section 1.1). Within these assumptions, several requirements (Section 1.2) and challenges (Section 1.3) arise towards developing the required solutions. The contributions of this thesis (Section 1.4) address these challenges by enabling high-fidelity reconstruction through multiple robots while maintaining communication efficiency.

1.1 Assumptions

A1 *Each robot is perfectly localized in an inertial frame of reference*
The contributions in this thesis target two (planning and mapping) of the three major subsystems (planning, mapping, and localization) of reconstruction frameworks. We assume a perfect localization system is available. This is a strong assumption since most existing state estimation methodologies accumulate drift in state estimates in visually challenging environments [59]. Potential integrations with relevant Simultaneous Localization and Mapping (SLAM) methods are suggested throughout the thesis when deemed necessary.

A2 *Each robot can follow forward-arc motion primitives*
The planning methodology in this thesis is based on forward-arc motion primitives-based control of mobile robots [146]. This is not a strong assumption since

planners have been proposed that utilize the same parameterization of motion primitives for heterogeneous robots [204, 47, 49]. Further, we note that fixed-wing aerial robots can also be used within a motion primitives-based framework [109, 169] with the exception that stopping maneuvers must be carefully designed [142]. For simplicity in exposition, we develop and demonstrate planning for multirotor vehicles in this thesis.

A3 *Each robot generates registered depth-intensity point clouds*
The mapping methodology in this thesis uses registered, four-dimensional (4D), depth-intensity point clouds as input data. This is not a strong assumption because most volumetric mapping approaches use at least three-dimensional depth data. Semantic or thermal mapping works use 4D data [177, 214]. For simplicity in exposition, we develop and demonstrate mapping for robots equipped with a forward-facing RGB-D sensor in this thesis.

A4 *The reconstruction environment is 3D and static*
Dynamic environments affect all subsystems of reconstruction frameworks and are out of scope for this thesis. Our primary target environments for deployments are complex caves and narrow industrial tunnels.

1.2 Requirements

R1 *Motion plans should be available anytime*
Faster replanning rates can lead to higher speed operation of robots in unknown environments [93]. However, calculating informative planning objectives can be computationally expensive depending on the map resolution and the sensing range [10, 134]. Different replanning rates might have to be set for every robot in the team depending on the compute capability. This requires the planning algorithm to be *anytime*, i.e., being able to return a plan within a fixed time budget.

R2 *Motion plans must be kinodynamically feasible*
Actuator constraints impose limits on the maximum achievable velocity and its higher derivatives given a starting state of the robot [125]. Therefore, the planning algorithm should allow for kinodynamic feasibility checks during informative planning to ensure safe operation during reconstruction.

R3 *Motion plans should adapt with environmental complexity*
A robot can potentially move much faster in wide open spaces compared to constrained tight spaces [149]. Further, it might need to move slower in regions with a high amount of surface detail to capture the map correctly compared to moving in regions away from any surfaces [134]. Hence, the maximum speed at any time during reconstruction should depend on the local environmental complexity. Note that this form of adaptation also reduces the number of maximum speed parameters that need to be set for a heterogeneous team deployment.

R4 *Map resolution should adapt with scene complexity*
Environments are unknown *a priori* for reconstruction tasks. It is undesirable to set a fixed map resolution throughout the reconstruction mission because this might lead to loss of fidelity on surfaces with a high amount of detail. Requiring map resolution to adapt with the scene complexity enables automatic modulation in the level of detail at which surfaces are captured in the map.

R5 *Map should be succinct for communication-efficiency*
In challenging environments, such as caves, the robots have to communicate over low-bandwidth communication channels [59]. Map representations are nominally the most expensive data structures to communicate amongst robots and to the human operators at some base station. In existing methods, this is achieved at the cost of fidelity of the map. However, a high-fidelity compressed map is desirable. Hence, the map representations used for the reconstruction framework should be communication-efficient while remaining high-fidelity.

R6 *Robots must not collide with each other or with the environment*
The planning algorithm must guarantee obstacle avoidance between the robots and the environment and should never enter into an inevitable collision state [93] during operation. Further, collision avoidance amongst the robots must occur via decentralized decision-making.

R7 *Informative planning using motion primitives without global occupancy grids*
Maintaining global occupancy grids can take a lot of memory onboard computationally constrained robots. Thus, we require enabling informative planning for multi-robot system using a compressed map representation.

1.3 Challenges

Several challenges need to be overcome in order to realize these requirements into one active reconstruction framework. The main challenges associated with the requirements are as follows:

C1 *Anytime and kinodynamically feasible motion planning*
Informative planning objectives are often expensive to compute as they require many ray casting operations [42]. Since the ray casting operations have an undecidable computational time complexity, it is challenging to serve suboptimal motion plans in an anytime manner (Requirement **R1**). Checking these suboptimal plans for kinodynamic feasibility (Requirement **R2**) entails careful motion plan generation and selection while satisfying maximum acceleration and jerk constraints.

C2 *Adaptive motion plans for efficient action selection*
How should the design of candidate motion plans vary with the environment clutter (Requirement **R3**)? The environment clutter is captured in the map

Chapter	Challenges	Publications
Chapter 4	**C1**	Goel et al. [73]
Chapter 5	**C2**	Goel et al. [75][1]
Chapter 6	**C3**	Goel et al. [76], Goel and Tabib [71]
Chapter 7	**C4**	Goel and Tabib [72]
Chapter 8	**C5**	Goel et al. [74], Tabib et al. [180][2]

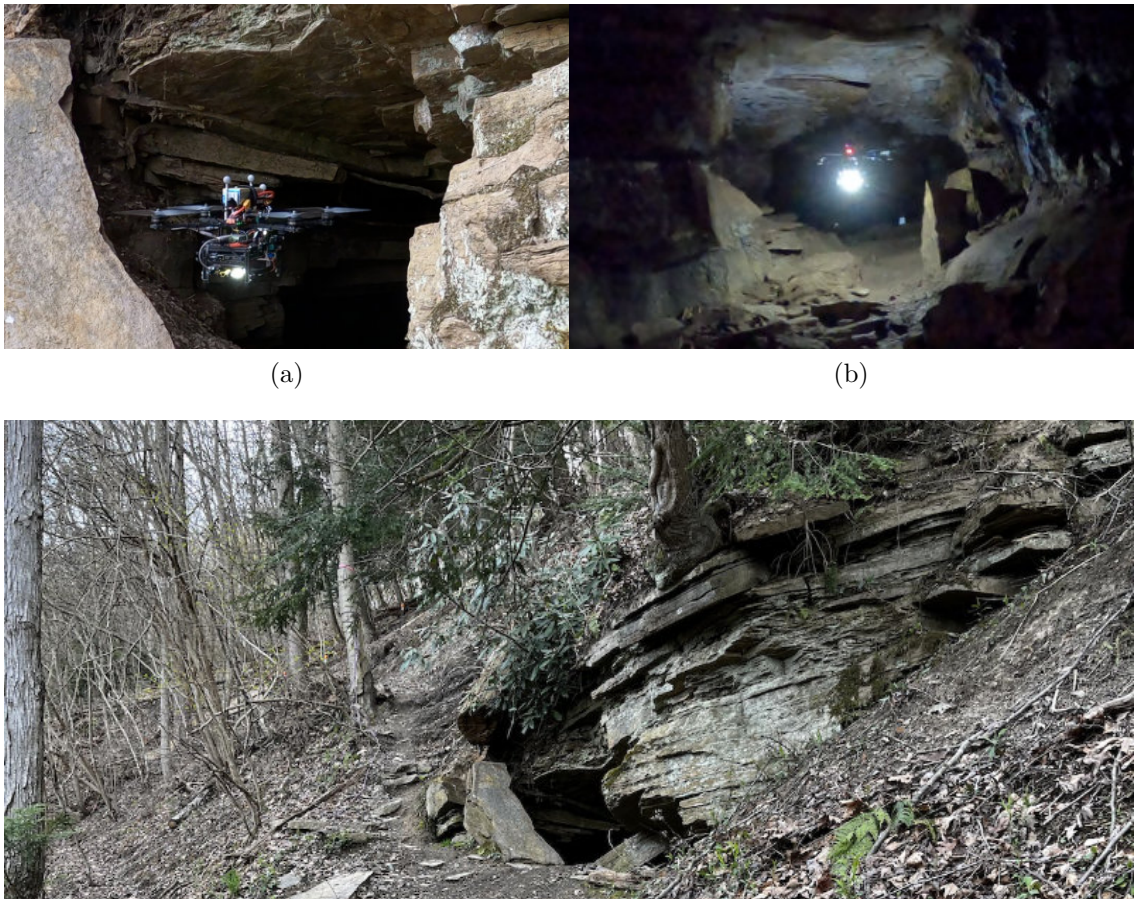
Table 1.1: The challenges addressed in this thesis by each chapter along with the associated publications.

since the environment is *a priori* unknown. The challenge here is to use some form of feedback from the map and adapt the design of possible motion plans.

C3 *Adaptive point cloud compression for high-fidelity mapping*
In this thesis, robots are equipped with RGB-D cameras that produce dense point clouds. Over time these point clouds need to be fused into a succint representation that can be used for downstream tasks such as reconstruction, map sharing, and safe informative planning (Requirements **R4**–**R7**). The key challenge is to automatically tradeoff the level of compression and the downstream task accuracy with changes in the scene.

C4 *Collision avoidance using compressed surface maps*
If a multi-robot team is creating compressed surface maps using onboard depth sensors, it follows that the same maps are used for collision avoidance (Requirement **R6**). However, to enable navigation in cluttered environments, it becomes crucial to avoid conservative approximations of the robot body and the environment. Therefore, the challenge here is to create compressed surface maps in a way that enables using tight approximations of robot bodies and the environment for collision avoidance.

C5 *Informative planning using compressed surface maps*
Existing map representations either use dense occupancy grids or raw point cloud data to enable informative planning for a multi-robot team during active reconstruction. If we are enabling collision avoidance using compressed maps, it is worth addressing how to enable informative planning with the same maps to avoid maintaining a separate map just for informative planning. The challenge is again to enable automatic tradeoff between the level of compression and the suboptimality of informative plans.



(a) (b)

(c)

Figure 1.1: Using the methodology in Chapter 5, a teleoperated multirotor adapts the motion planning speed and local map resolution to (a) enter a cave and (b) traverse a tight passage inside. (c) illustrates the surroundings near the cave entrance, which is embedded in a sloping hillside. A video of this experiment can be found at https://youtu.be/VjyoPVXT8WY.

1.4 Contributions & Outline

Towards solving these challenges, this thesis makes the following contributions (Table 1.1).

Chapter 2 provides a literature review of planning and mapping for robotic reconstruction and highlights the research gaps addressed through this thesis.

Chapter 3 presents background information such as preliminary concepts in motion planning and spatial mapping, the multi-robot reconstruction problem, and Gaussian mixture models for mapping.

[1]Best Paper Award, IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2022

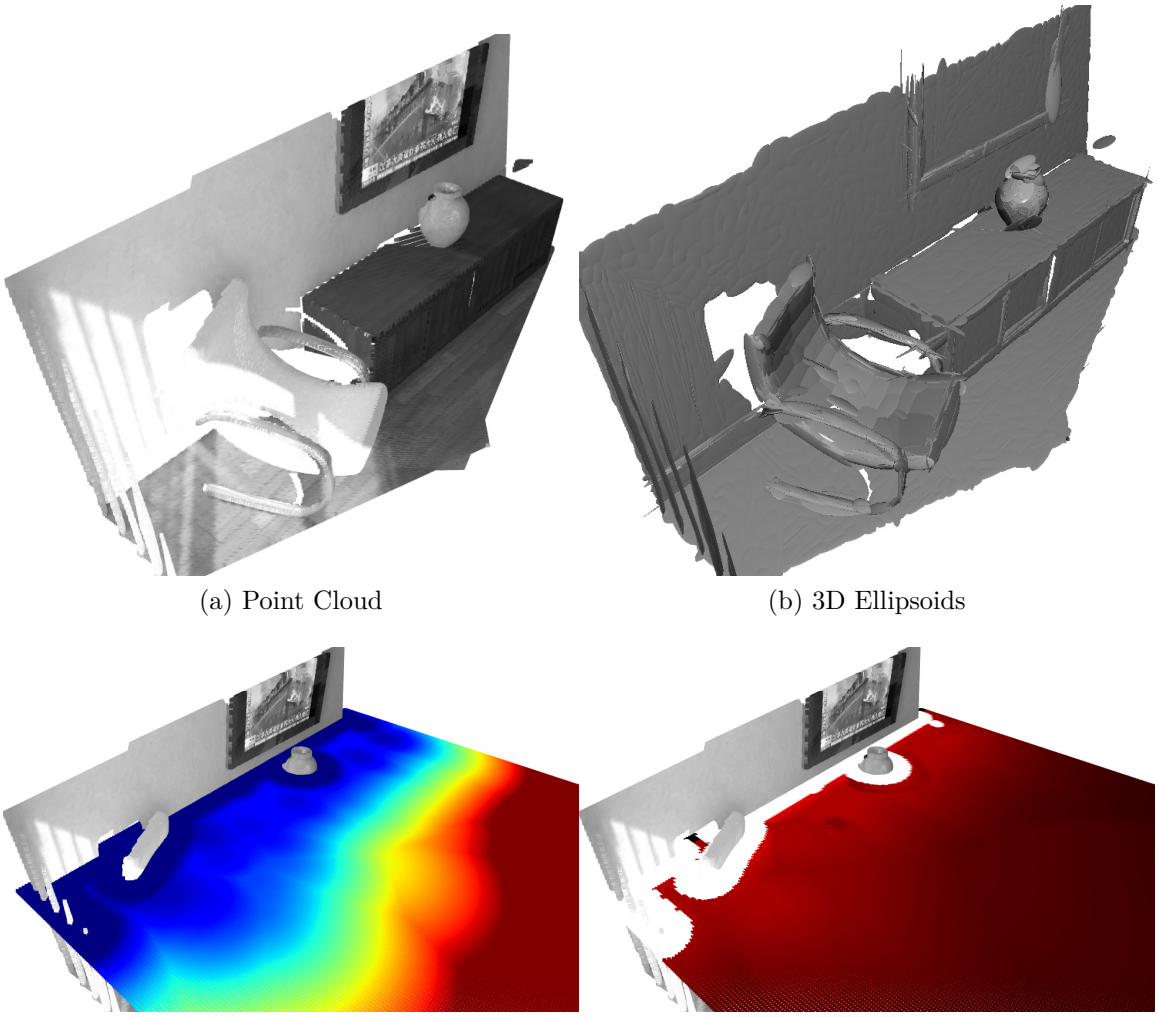
[2]King-Sun Fu Memorial Best Paper Award (Honorable Mention), IEEE Transactions on Robotics, 2022

Figure 1.2: The methodology proposed in Chapter 6 enables multi-modal reconstruction at varying scales. Without modifying the model complexity parameters, the methodology models depth and grayscale data of small objects ($1\,\text{m} \times 1\,\text{m}$ safety cone in the left image) and complex environments ($10\,\text{m} \times 5\,\text{m}$ cave in the center image) while also modeling depth and thermal data of (right) large-scale buildings ($42\,\text{m} \times 28\,\text{m}$) [18]. A supplementary video may be found at https://youtu.be/TCn5KB1m5P0.

Chapter 4 [**73**] addresses challenge **C1** through a motion primitives-based planner that utilizes a steady-state analysis of maximum feasible speeds under sensing and kinodynamic constraints to design the library of motion primitives (MPL) along with a Monte Carlo tree search (MCTS) strategy for motion primitive selection. The method is tested onboard a multirotor which achieves speeds exceeding $2.25\,\text{m/s}$ during reconstruction in open spaces and around scattered obstacles while enabling anytime and safe operation.

Chapter 5 [**75**] targets challenge **C2** and presents a hierarchical collision avoidance methodology that automatically initializes and modulates the maximum speed parameter based on the environmental complexity (Fig. 1.1). The map resolution of a local occupancy grid is adapted hierarchically using the feedback from a local distance field-based collision checker. If the number of 3D voxels in the local map is fixed, it is shown that in this case the maximum speed parameter is a function of the map resolution. The results from simulated and real-world cave reconstruction scenarios demonstrate automatic speed modulation as the multirotor navigates through regions with diverse environment complexity. Further, since the map resolution is adapted online, teleoperation through tight spaces becomes feasible in scenarios where it otherwise would be infeasible if map resolution were fixed.

Chapter 6 [**76, 71**] contributes an information-theoretic mapping method, Self Organizing Gaussian Mixture Modeling (SOGMM), to address challenge **C3**. Through this technique, the number of components of a GMM-based mapping methodology are adapted during operation based on the complexity of depth-intensity image pairs (Fig. 1.2). The principle of relevant information (PRI) [147] is employed to estimate the relevant modes in the image data. The number of these relevant modes is used as the number of components for a GMM over the 4D data (3D point cloud and 1D intensity information per point). The qualitative and quantitative results demonstrate the adaptation of number of components with changing scene complexity on real-world datasets; scenes of increasing depth-intensity complexity are automatically assigned higher number of components.



(a) Point Cloud (b) 3D Ellipsoids

(c) Euclidean Distance Field (d) Collision Probability

Figure 1.3: Chapter 7 contributes methods to estimate continuous-space collision probability, Euclidean distance and gradient of an ellipsoidal robot body model from a Gaussian surface model (GSM) of the surface. The 3D point cloud shown in (a) is approximated with a GSM, shown as a set of ellipsoids in (b). (c) The Euclidean distance over a 2D slice predicted by the proposed approach is shown as a heatmap (increasing distances from blue to red). (d) The collision probability values (decreasing from red to black, 1.0 in white regions) over the same 2D slice when the robot position is uncertain.

Chapter 7 [**72**] studies the problem of estimating distance and collision probability between a set of ellipsoids towards enabling collision avoidance between multiple robots and between robots and the environment (Challenge **C4**, Fig. 1.3). Most collision detection and avoidance approaches assume the robot is modeled as a sphere, but ellipsoidal representations provide tighter approximations and enable navigation in cluttered and narrow spaces. State-of-the-art methods derive the Euclidean distance and gradient by processing raw point clouds, which is computationally expensive for large workspaces. Gaussian mixture surface mapping enables compressed and high-fidelity surface representations. Few methods exist to estimate continuous-space occupancy from such models. They require Gaussians to model free space and are unable to estimate the collision probability, Euclidean distance and gradient for an ellipsoidal robot. The proposed methods bridge this gap by extending prior work in ellipsoid-to-ellipsoid Euclidean distance and collision probability estimation to Gaussian surface models. A geometric blending approach is also proposed to improve collision probability estimation. The approaches are evaluated with numerical 2D and 3D experiments using real-world point cloud data. Methods for efficient calculation of these quantities are demonstrated to execute within a few microseconds per ellipsoid pair on modern embedded computers.

Chapter 8 [**74, 180**] improves upon prior work where GMM-based maps are used for reconstruction [43], and details a real-time distributed mapping system that represents the global shared map representation as a GMM. The occupancy information is locally constructed from the GMM map [138] and used for information-theoretic planning [179] (Challenge **C5**). The results from simulation experiments suggest improvement in reconstruction performance on constrained bandwidth channels as compared to occupancy grids. In the real-world, a two-multirotor system is deployed in a wild cave. We observe up to 100x improvement in the communication-efficiency compared to discretized mapping techniques.

In **Chapter 9**, we state the concluding remarks along with a summary of results obtained through this thesis. Promising directions of future work are also discussed.

CHAPTER 2
Related Work

This chapter reviews prior work in adaptive-anytime planning and mapping towards addressing challenges **C1**–**C5**. We limit the scope of this review to works that present solutions under the same assumptions (Section 1.1). For **C1** and **C5**, safe and anytime informative planners for 3D reconstruction are reviewed in Sect. 2.1. For **C2**, the related work in adaptive multirotor navigation is presented in Sect. 2.2. For **C3**, adaptive point cloud compression methods that are useful for 3D mapping in reconstruction applications are reviewed in Sect. 2.3. For **C4**, we review methods that provide Euclidean distance and gradient calculation in the continuous space (Sect. 2.4).

2.1 Safe, Anytime Informative Planning for 3D reconstruction

Motion primitives-based reconstruction for existing reconstruction systems [49, 141, 209, 25, 46, 58] typically consists of two stages: action representation and action selection. Action representation refers to the motion profile used by the planner to represent a motion to a potentially informative region in the 3D space. Action selection is a two-stage process. First, a utility function to calculate the information the candidate action can provide about the environment. And second, a spatial sampling of the candidate actions to search the explored map so far for potential, reachable, informative spaces. The second stage is what usually enables anytime operation [43, 9, 13]. Next we review the literature in each of these areas.

Action Representation. Cieslewski et al. [38] propose a strategy based on maintaining rapid forward motion by driving the system toward frontier cells (cells on boundary of free and unknown space [201]) within the camera field-of-view. The frontier-based approach is the most common amongst the deployed reconstruction frameworks. While the authors note that reaction time for obstacles avoidance can limit speeds in reconstruction, they provide little discussion of why this happens or how it can be avoided. In contrast, the action representation developed through this thesis considers a broader variety of sensing actions that can avoid these limitations and also incorporates sensing and planning time into action design. In this thesis, through Chapter 4 we focus on design of a library of motion primitives suitable for reconstruction at high speeds given the relationship between sensing constraints and maximum safe velocities.

Action Selection. The selection strategy requires creating a utility function that

can be calculated over the candidate actions. Cieslewski et al. [38] use a frontier-based motion planning strategy in which the planner biases the motion plans towards frontiers while penalizing a change in direction of velocity. Recent work by Cao et al. [26] improves upon frontier-based coverage strategies and proposes a hierarchical planner that uses surface normals to generate informative viewpoints via a geometric heuristic. These objectives can provide a rapid coverage of the environment with low-noise sensors such as LiDARs. However, since the uncertainty in the map is not explicitly accounted into motion planning, maximizing coverage alone might result in degraded performance when using noisy data from depth sensors.

Towards incorporating uncertainty of the map explicitly into decision making, there is a line of research that uses mutual information between projected measurements and the current map as the objective to drive reconstruction. However, these methods are computationally expensive when compared with coverage methods. Charrow et al. [30] and Zhang et al. [207] propose a computationally tractable approximation of the Cauchy-Schwarz Quadratic Mutual Information (CSQMI) and Shannon Mutual Information (FSMI) respectively, and use this information metric over a local lattice of feasible trajectories to greedily maximize information in a local map. The problem with local and greedy approaches is that they are susceptible to getting stuck in local extrema of information distribution over the environment. To this end, modifications have been proposed to these information-theoretic strategies that incorporate not only a local information maximization objective, but also allows to reason over the global information distribution. Approaches by Charrow et al. [29] and Tabib et al. [179] use frontiers to model this global information spread. For our reconstruction planner in Chapter 4 we use the objective function by Corah et al. [43] which uses a global library of informative views as a succinct and more accurate representation of the global information content.

Anytime Operation. The objective of anytime planning for reconstruction is to be able to compute plans given a time budget and the current information of the state space [162]. To enforce anytime operation, receding-horizon approaches are used in 3D reconstruction scenarios [13, 43]. In Chapter 4, we employ Monte Carlo tree search (MCTS) as it can search over long planning horizons using computationally expensive reward functions, and it is anytime [21]. Further, MCTS is parallelizable on the CPU and GPU [32] and thus it is beneficial towards enabling anytime planning and mapping for the target platforms considered in this thesis.

2.2 Adaptive Planning for 3D Navigation

Motion primitives-based teleoperation of a multirotor has been demonstrated to reduce operator cognitive load [203]. Spitzer et al. [165] propose a teleoperation approach that leverages a KD-Tree to locally represent the environment for fast collision checking. A motion primitive pruning approach leverages low-latency collision avoidance and the closest distance to the operator's joystick input to allow for adaptive-speed teleoperation in the presence of obstacles. The local environment representation assumes the resolution of the map is fixed before the vehicle

is teleoperated and does not encode unknown or free space information. However, in practice, the unknown space information may be necessary to ensure safety when the environment is incrementally revealed through a limited range depth sensor [93, 186]. Free space information is necessary to simultaneously support applications such as robotic reconstruction [180]. Thus, in this thesis, we use probabilistic occupancy maps for local environment representation. A fixed resolution during operation imposes restrictions on the configuration space for the planner because the motion primitive pruning approach depends on the granularity of discretization. If the resolution is too low, the robot might not be able to enter narrow entrances due to the coarseness of the map. If the resolution is too high, the high perceptual latency of a fine map limits the maximum speed of the robot [64]. In this thesis, we address this research gap by proposing a multirotor teleoperation approach, which uses variable-resolution local probabilistic occupancy maps hierarchically to enable fast teleoperation in open spaces while ensuring safe, low-speed teleoperation through narrow passages.

Prior work in adaptive motion planning has been proposed to modulate robot speed based on application-specific heuristics. Zhang et al. [205] present a likelihood-based collision avoidance strategy for fast teleoperation of a multirotor by prioritizing open spaces for navigation to maintain high speed. The objective assumes that taking an alternate path (i.e., a path through open space) will lead to the same location as taking another (i.e., a path through a narrow passage); however, this assumption is flawed in the context of certain domains like caves. In contrast, our work adapts the maximum vehicle speed according to environment complexity. Quan et al. [149] propose an adaptive optimization-based motion planning approach for the multirotor navigation task, which is most similar to our approach. The heuristic used for speed modulation relies on the angle between the velocity direction and the gradient of the local signed distance field. This heuristic is coupled with a multi-layer model predictive control approach to allow for fast flight in sparsely cluttered environments and slow flight through dense clutter. However, while the motion planner allows adaptation in speed, the resolution of the environment representation is fixed. Selecting the resolution of an *a priori* unknown environment is difficult, which makes the approach of Quan et al. [149] unsuitable for our application. An alternative strategy is to operate with a range of voxel sizes for the local environment representation and modulate speeds according to the change in the map. The proposed approach builds on this strategy and introduces a motion primitive selection approach that modulates the maximum speed along the motion primitives according to the voxel size of the local occupancy map.

Several methods exist for hierarchical volumetric occupancy mapping. OctoMap by Hornung et al. [88] provides a multi-level representation of occupancy via an Oc-Tree data structure. However, to the best of our knowledge, no motion planner exists that can leverage multiple levels of a local OctoMap-based representation to allow for adaptive-speed teleoperation through environments with varying clutter. Nelson et al. [134] present an occupancy grid adaptation methodology for two-dimensional environment reconstruction using ground robots. This approach is specific to the reconstruction scenario and the voxel sizes are adapted via the information-bottleneck method to minimize the cost of computing the reconstruction objective. It is unclear

how this approach may be applied in the context of multirotor teleoperation since the objective highly depends on the intent of the human operator and may vary over time. Closest to our work, Funk et al. [68] create a multi-resolution OcTree-based representation for accurate 3D reconstruction and online planning. The resolution is selected by minimizing the error in occupancy representation. A path planning result is presented in this thesis that utilizes a “coarse-to-fine" approach for collision checking. The time to compute the motion plan varies from 0.01 s to 0.4 s depending on the desired start and end-points and the maximum resolution set by the user. However, the results are generated with post-processed datasets and a parallelized implementation so it is unclear how the performance would translate to compute constrained robotic systems and real-world results. A key consideration in the teleoperation context is to minimize the lag in the multirotor response felt by the operator by having a high motion planning rate. Thus, it is important to design a planning approach that maintains a steady planning rate in environments with varying clutter. The proposed approach addresses this gap by adapting a pre-specified number of map levels per planning round to achieve a planning rate of at least 10 Hz.

2.3 Adaptive Point Cloud Compression

In this section, we focus on reviewing methods that enable adaptive point cloud compression for 3D mapping purposes. Both discrete and continuous methods are reviewed.

Discrete Methods. Robotics applications commonly leverage probabilistic environment representations consisting of fixed-size volumetric pixels (or voxels in 3D) and represent occupancy as independent, discrete random variables [63]. The discrete probability distribution is updated using the inverse sensor model and Bayes' rule. The limitation of this representation is it requires specifying parameters such as voxel size and occupancy clamping thresholds, which are dependent on environmental conditions. Although the compute usage for Bayesian updates is low, voxel grids require significant memory to scale in spatially-extensive environments. Multi-modal extensions for 3D voxel grids suffer from the same limitations [177].

To reduce the number of voxels, Magnusson et al. [118] propose the Normal Distributions Transform mapping framework (NDTMap) that utilizes the same volumetric factorization but places a Gaussian distribution in each occupied voxel. The Voxblox method by Oleynikova et al. [137] uses truncated signed distance functions (TSDFs) over regular 3D voxel grids for point cloud modeling as opposed to an occupancy-based formulation. Both of these frameworks model the point cloud with less memory than regular 3D voxel grids, but still require specifying a map resolution in advance.

Hornung et al. [88] present OctoMap to address the limitation regarding pre-specified voxel resolution via the hierarchy of an octree. The limitations are that pre-specifying a minimum leaf size, which determines the maximum size of the map, and occupancy thresholds are still required. Recent advances over OctoMap demonstrate improvements in compression and fidelity [57, 68]; however, the limitations regarding pre-specified parameters are unaddressed.

In contrast to these discrete methods, we present a continuous probabilistic multi-modal modeling methodology that adapts the model complexity according to the scene variation.

Continuous Methods. Continuous non-parametric point cloud modeling methods like Gaussian Process (GP) [136] and Hilbert maps [151] increase representational power at the expense of high computational cost. Closest to our work, Zobeidi et al. [214] develop an incremental metric-semantic GP mapping approach that represents map uncertainty and outperforms deep neural network-based approaches. However, the need for offline training and specification of model hyperparameters precludes the use of this representation for online adaptive compression. Yan et al. [202] present a highly-parallelized adaptive scene representation that utilizes a Dirichlet Process (DP) mixture model as a point cloud model. However, an upper limit on the number of components allowed per processor must be pre-specified, which limits the adaptability of the resulting model.

Neural Implicit Representations (NIRs) have been proposed as continuous implicit models for radiance and occupancy information. Most works in this area build on Neural Radiance Fields (NeRFs) proposed by Mildenhall et al. [127]. While NeRFs produce high-resolution, realistic models, they suffer from one major limitation due to the fixed architecture of the underlying neural network, which does not allow adaptivity in the representation of the scene.

Gaussian Mixture Models for Mapping. Generative probabilistic models that use a finite mixture of probability distributions aim to represent the environment through an adaptive and parametric mathematical model, and thus provide a potential solution to the limitation in adaptivity of NeRFs. Recently, Gaussian mixture models (GMMs) have been utilized for adaptive point cloud modeling [61, 133, 167, 138, 55]. Eckart et al. [61] use a top-down hierarchy of 3D GMMs to represent 3D point cloud data at different levels of detail. Srivastava and Michael [167] use a bottom-up hierarchy of 4D GMMs to model the surface point clouds and the points along the sensor beams. Further, these works utilize random initialization for the hierarchical GMMs that does not yield accurate results [113]. Navarrete et al. [133] modify the FastGMM framework by Greggio et al. [77] and use it for compressing locally planar point clouds. The FastGMM approach creates a GMM via a deterministic splitting criterion in a coarse-to-fine manner. Dong et al. [55] use a similar idea for adaptive model selection in their GMM mapping approach. The Gaussian components belonging to the same plane are merged using their mean vectors and principal directions to obtain a coarser model. Common to all these works is a requirement to specify the levels of detail; either directly (e.g., number of layers in hierarchy of GMMs) [61] or indirectly (e.g., through convergence, splitting, or merging criteria) [167, 133, 55]. Specifying these parameters *a priori* can be challenging for real-world robotic applications. In contrast to these methods, the proposed approach does not require processing a hierarchy of GMMs to achieve online adaptation with respect to scene complexity.

The fundamental challenge is to estimate the number of components required to model the point cloud to obtain sufficient fidelity. McLachlan and Rathnayake [123] provide an overview of approaches to estimate the number of components in

a GMM. Among the commonly used criteria are the Akaike's information criterion (AIC) [4], Bayesian information criterion (BIC) [163], and Model description length (MDL) [82], which strike a balance between the fit over the underlying dataset and model complexity. For example, to use AIC or BIC scores for model selection one must plot the scores over many candidate mixtures with increasing numbers of components to detect an approximate minima, which is prohibitive for online robotics applications with finite computational power and timing constraints. Variational methods have been created to estimate *a posteriori* distributions over the parameters of a GMM; however, these methods also require specifying a prior distribution over model parameters [7, 17]. As opposed to using these techniques, we perform model selection for GMMs through information-theoretic learning.

The proposed approach also relates to Self-Organizing Maps (SOMs) by Kohonen [102] through the Self-Organizing Mixture Models (SOMMs) [189]. To create SOMMs, Verbeek et al. [189] present a modified Expectation Maximization (EM) algorithm where the E-step enforces topology preservation while the likelihood of the data is maximized. Consequently, this approach provides topology-preserving fits of the data. However, as the authors point out in their concluding remarks, setting the number of components in their approach is a non-trivial task. The point cloud modeling method proposed in our work (Chapter 6) is able to estimate the number of components adaptively.

Saarinen et al. [160] motivate the development of NDTMap, which uses a Gaussian density in each cell, by arguing that larger voxels may be used since the Gaussian density better approximates the surface geometry. However, this representation also suffers from the aliasing challenges of a regular occupancy grid as each Gaussian density is considered an independent component of a uniformly-weighted Gaussian mixture model (GMM). [43, 179] relax the assumption of uniform weights, by using a maximum-likelihood fit over the point cloud data to create a global map that is represented as a GMM without the use of a discrete grid. However, these works require specifying the number of mixture components before operation which limits the maximum achievable fidelity of the map. We bridge this gap by proposing a GMM-based approach that enables creating a map representation that increases the model fidelity incrementally via an information-theoretic self-organizing approach [76] and enables scalable and efficient inference via spatial hashing.

Neural Radiance Fields (NeRFs) [127] enable photorealistic environment rendering at lower memory costs; however, incremental mapping with implicit representations are known to suffer from catastrophic forgetting [208]. Catastrophic forgetting is the problem of forgetting old knowledge after training with new data [122]. To mitigate this issue, some incremental NeRF mapping approaches [172, 139] retain keyframes from historical data and replay them with current data to train the network; however, this approach requires more memory to store the keyframes [208]. Zhong et al. [208] develop a technique for large scale Signed Distance Field (SDF) mapping using LiDAR, but it is not clear how robust it will be to catastrophic forgetting when intensity data is incorporated. In contrast, the proposed approach adaptively increases the fidelity of the parametric GMM-based environment model depending on the scene complexity. This way, while offering an implicit representation of the surface, no

special consideration for catastrophic forgetting is required.

While it may be possible to use NeRFs for the reconstruction problem [161], it is unclear how to use this representation for map sharing in multi-robot reconstruction. In contrast, GMM-based maps can provide the sensor pose locally [178] and globally [176], can be used for multi-robot reconstruction [74], and enable adaptive multimodal reconstruction through the proposed approach built on top of SOGMMs [76]. Zhu et al. [213] propose NICE-SLAM to map multiple modalities. But the representation suffers from memory inefficiency due to the use of multiple 3D voxel grids. The proposed approach uses only a sparse grid over the Gaussian components for computation performance gains for incremental mapping. Finally, note that the current state-of-the-art in radiance field rendering in the computer graphics literature leverages 3D Gaussian densities in place of neural networks [99, 98], demonstrating superior performance in both training and inference compared to NeRFs. Our method uses a mixture of 4D Gaussian densities to jointly model intensity and spatial data. Therefore, incorporating radiance field rendering within the proposed incremental mapping method is an exciting direction for future work. There are several concurrent works that use the depth data [97, 173] but still require offboard compute and do not enable motion planning and active informative planning. Some recent works do propose active informative planning solutions that require the planner to not be anytime or without safety guarantees [95, 197]. Works that do support safety guarantees [33] are not amenable to informative planning. Works that claim to achieve both, such as the RT-GuiDE [181], are not amenable to multi-robot information sharing due to the large number of Gaussians utilized for reconstruction. In contrast, the proposed work enables high-fidelity multi-robot active reconstruction while enabling safe collision-free and kinodynamically-feasible operation of the individual robots in the team.

2.4 Continuous Space Collision Avoidance

Euclidean distance fields and their gradients (often computed using finite differencing) are used for collision avoidance [137] and optimization-based motion planning [154], respectively. These values are calculated by fusing point clouds acquired by range sensors (e.g., LiDARs and depth cameras) and processed on-board the robot. To enable safe navigation outside the sensor field-of-view (FoV), the robot must maintain a map comprised of past fused observations. This is especially important when limited FoV sensors, like depth cameras, are used onboard the robot [186]. Existing Euclidean distance and gradient estimation methods create a spatially discretized map and compute distance and gradient values in each cell via a Breadth-First Search [81], which is computationally expensive in large workspaces or when small grid cells are employed [34]. To overcome the limitations of cell based methods, Gaussian Process (GP) based methods were developed, which implicitly represent the surface and enable continuous-space Euclidean distance and gradient calculation [108, 196, 107]. A GP is regressed from raw point cloud data and used to infer the Euclidean distance and gradient at any test point in the workspace. A challenge with these methods is that the training time scales as the size of the input data increases, so optimization-

based subsampling must be introduced to enable spatial scalability [195]. In contrast, Gaussian surface models (GSMs) (e.g., mixture models [61, 167, 138, 110] and 3D splatting works [50, 99, 98]) provide relatively compressed and high-fidelity point cloud models. This work leverages the geometric interpretation of GSMs to calculate continuous-space Euclidean distance and gradients.

Collision probabilities are used for motion planning under robot position uncertainty [56, 22]. Like Euclidean distance and gradient field methods, existing methods that estimate collision probability from a surface point cloud utilize probabilistic occupancy queries from a discrete [86, 104] or continuous surface representation [66]. These maps require storing free space cells [63, 88, 3] or raw point clouds [151], leading to high memory usage. GSMs can alleviate these challenges by effectively compressing point clouds into a finite set of ellipsoids. However, existing occupancy estimation frameworks leveraging GSMs either fit Gaussian components in free space [167, 111] or Monte Carlo ray trace through a local discrete grid [138, 179]. Robot pose uncertainty is not considered. This thesis bridges these gaps by proposing a collision probability estimation method, which uses only the surface Gaussians and accounts for Gaussian uncertainty in the robot position.

All the aforementioned techniques require approximating the shape of the robot. Spherical approximations are widely used because inflating the obstacles with the robot size for the purposes of collision avoidance results in a simple configuration space (C-space) [105]. However, recent results in ellipsoidal approximations demonstrate superior planning performance, especially in cluttered and narrow environments [158] for both rigid [114] and articulated robots [94]. These approximations are also used for human bodies in a dynamic environment [212, 115]. Consequently, in this work the proposed methods use an ellipsoidal robot body approximation.

Gaussian Process regression has been used extensively in continuous EDF generation. Lee et al. [108] model the surface point cloud using a GP implicit surface (GPIS) [193], which enables creating continuous ESDFs that are limited to a short-range from the surface. Wu et al. [194] propose the Log-GPIS map representation that estimates continuous EDFs further into 2D and 3D workspaces. This method models heat diffusion from the surface [45] as a GP regression and uses the logarithm of the regressed heat as the distance. The gradient is also regressed as part of the GP regression. The error in EDF estimates is demonstrated to be lower than the popular discrete-space Voxblox [137] approach. The downside of this method is that the logarithm operation is numerically unstable when the heat approaches zero, which occurs at distances far from the surface. Le Gentil et al. [107] prove that the error in Log-GPIS EDF increases with the distance from the surface and propose an alternate formulation, which leverages reverting functions of covariance kernels for GP regression [153]. While this method achieves EDFs that are relatively accurate, the numerical instability limitation still exists because a logarithm operation is needed over an occupancy value that gets close to zero far from the surface. Furthermore, all of the GP-based methods assume a spherical robot body. In contrast, our approach is numerically stable far from the surface and explicitly accounts for ellipsoidal robot shapes.

GPIS-based representations provide a variance estimate, which indicates the un-

certainty in the fused map; however, there remains an open research question about how to compute a collision probability leveraging this variance and the robot position uncertainty. Our approach, however, does provide a collision probability estimate.

Few deep learning approaches enable EDF and gradient estimation. Ortiz et al. [140] model the signed distance function using a 4-layer multi-layer perceptron along with 3D positional embedding for input coordinates [127]. During training, an upper bound on the distance is calculated from sampled free space points by approximating the location of the closest surface points. In larger workspaces, this method requires replaying stored keyframes to avoid catastrophic forgetting. Just like GP-based methods, a spherical robot body is assumed implicitly. Instead of the closest point, we use the notion of the closest ellipsoid on the surface. This ellipsoid belongs to a set of ellipsoids that geometrically approximate the surface GSM. The GSM is created using a methodology that does not suffer from catastrophic forgetting and does not require free space data [71]. Note that the GP-based methods mentioned earlier also do not need to account for catastrophic forgetting and do not use free space points. Due to this reason, the state-of-the-art GP-based method in [107] is used as a baseline method for distance field and gradient comparison (Sect. 7.2).

Nguyen et al. [135] utilize deep neural networks to predict the collision probability using a depth image, uncertain partial robot state, and motion primitives as inputs. The network is trained in simulation environments using 1.5 million data points. The Gaussian uncertainty in robot state is propagated through the network through an Unscented Transform [96] of Monte Carlo samples of the Gaussian. The model uncertainty is obtained via the Monte Carlo dropout [69] approach. Both uncertainty calculation methods contain randomness because of the Monte Carlo sampling. Due to this randomness, it is unclear if the uncertainty measure will generalize to real-world environments that are significantly different from the simulation environments used for training. Furthermore, the training will also require data points obtained at various noise levels in robot state, increasing the cost of prior training. In contrast, the proposed method is geometric and does not require prior training.

There are a few methods to note that enable relevant applications without explicitly creating EDFs. Dhawale et al. [52] demonstrate reactive collision avoidance for a quadcopter from a Gaussian mixture model (GMM) surface model without creating EDFs. The Gaussian components of the GMM are approximated as iso-contour ellipsoids. To simplify the C-space [116], a spherical robot approximation is leveraged to inflate the ellipsoids by the radius of the robot body model. Given the inflated ellipsoids, the robot is approximated as a point object and point-in-ellipsoid tests are used for deterministic collision tests. For the same task, Florence et al. [65] provide collision probability estimates for a spherical robot body. Liu et al. [114] use an ellipsoidal model for a quadcopter and use the raw point cloud maps (within a KD-Tree for scalability) for point-in-ellipsoid tests. Srivastava and Michael [167] and Li et al. [111] use surface and free space GMMs to compute continuous-space occupancy prediction. However, the occupancy prediction does not encode robot position uncertainty so that a collision probability may be regressed. In contrast, the proposed methods enable continuous-space collision probability, Euclidean distance and gradient calculation without using free space GMMs for ellipsoidal robots.

CHAPTER 3
Background

This chapter provides an overview of two basic aspects used in the remainder of this thesis. The contents of this chapter also appear in [179, 180].

3.1 Preliminaries

3.1.1 Notation

In this thesis, unless otherwise noted, small letters are used for scalars and univariate random variables (e.g., w, h, x, g), capital letters for multivariate random variables (e.g., X, Y), bolded small letters for vectors (e.g., $\mathbf{x}$, $\mathbf{y}$), bolded capital letters for matrices (e.g., $\mathbf{X}$, $\mathbf{Y}$), and calligraphic letters for sets (e.g., $\mathcal{I}$, $\mathcal{X}$, $\mathcal{Y}$). The probability density of a continuous multivariate random variable X is written as $p_X(\mathbf{x})$, where $\mathbf{x}$ is a value in the sample space of X.

3.1.2 Gaussian Mixture Models

A GMM represents D-dimensional multivariate data ($D \in \mathbb{Z}_{>0}$) as a weighted combination of M Gaussian densities [15, p. 110]. The probability density function of a GMM is parameterized by a set $\Theta = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^{M}$, where $\pi \in \mathbb{R}$ is a weight and $\sum_m \pi_m = 1$. The probability density of the GMM is written as

$$p(\mathbf{x} \mid \Theta) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{3.1}$$

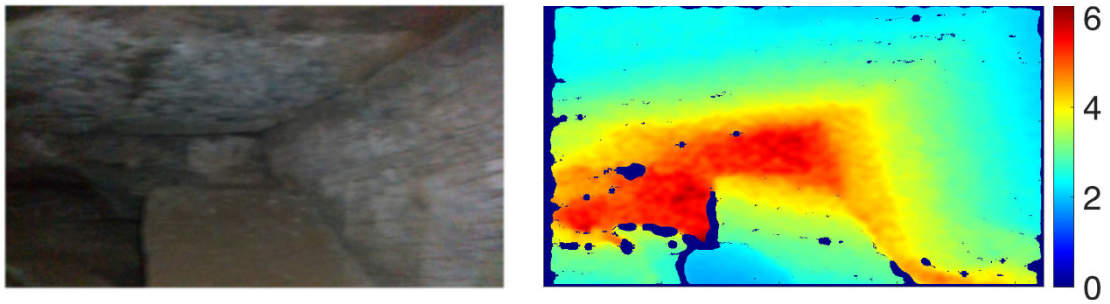
where $\mathbf{x} \in \mathbb{R}^D$ is an independent and identically distributed (i.i.d) sample drawn with distribution p, and

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_m)^\top \boldsymbol{\Sigma}_m^{-1}(\mathbf{x} - \boldsymbol{\mu}_m)\right)}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_m|^{1/2}} \tag{3.2}$$

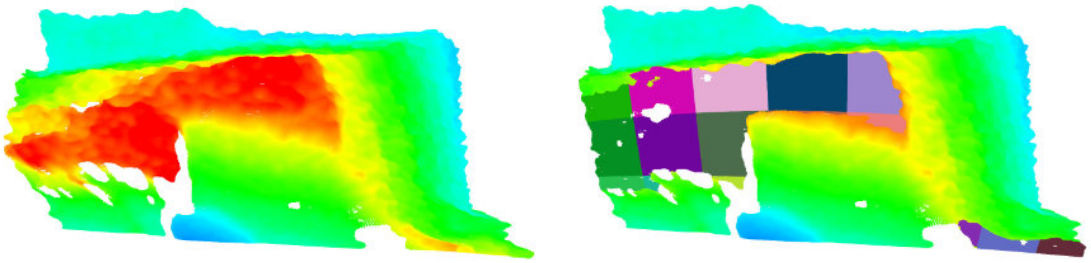
is the probability density function of a Gaussian distribution with mean $\boldsymbol{\mu}_m \in \mathbb{R}^D$ and covariance $\boldsymbol{\Sigma}_m \in \mathbb{R}^{D \times D}$.

3.2 Gaussian Mixture Models for reconstruction

Gaussian Mixture Models (GMMs) can be leveraged to compactly encode sensor observations for transmission over low-bandwidth communications channels [179]. The



(a) Color Image (b) Depth Image



(c) Point Cloud (d) Free Space Windows

(e) GMM (f) Resampled Data

Figure 3.1: Overview of the approach to transform a sensor observation into free and occupied GMMs. (a) A color image taken onboard the robot exploring Laurel Caverns. (b) A depth image corresponding to the same view as the color image with distance shown as a heatmap on the right hand side (in meters). (c) illustrates the point cloud representation of the depth image. (d) In the mapping approach, points at a distance smaller than a user-specified max range r_d (in this case $r_d = 5\,\text{m}$) are considered to be occupied, and a GMM is learned using the approach detailed in Sect. 3.2.1. Points at a distance further than r_d are considered free, normalized to a unit vector, and projected to r_d. The free space points are projected to image space and windowed using the technique detailed in Sect. 3.2.2 to decrease computation time. Each window is shown in a different color. (e) The GMM representing the occupied-space points is shown in red and the GMM representing the free space points is shown in black. Sampling 2×10^5 points from the distribution yields the result shown in (f). The number of points to resample is selected for illustration purposes and to highlight that the resampling process yields a map reconstruction with an arbitrary number of points.

GMM provides a generative model of the sensor observations from which occupancy may be reconstructed by resampling from the distribution and raytracing through a local occupancy grid map. Formally, the GMM is a weighted sum of M Gaussian probability density functions (PDFs).

A depth observation taken at time t and consisting of N points,

$$\mathcal{Z}_t = \{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^n, \ldots, \mathbf{x}_t^N\},$$

is used to learn a GMM. The Expectation Maximization (EM) algorithm is usually employed to solve the maximum-likelihood parameter estimation problem, which is guaranteed to find a local maximum of the log likelihood function [15].

Following the work of O'Meadhra et al. [138], distinct occupied $\mathcal{G}(\boldsymbol{x})$ (detailed in Sect. 3.2.1) and free $\mathcal{F}(\boldsymbol{x})$ (detailed in Sect. 3.2.2) GMMs are learned to compactly represent the density of points observed in the environment (Fig. 3.1). The process by which $\mathcal{F}(\boldsymbol{x})$ and $\mathcal{G}(\boldsymbol{x})$ are created is illustrated in Figs. 3.1c and 3.1d. Because the GMM is a generative model, one may sample from the distribution (Fig. 3.1f) to generate points associated with the surface model and reconstruct occupancy (detailed in Sect. 3.2.3).

3.2.1 Occupied Space

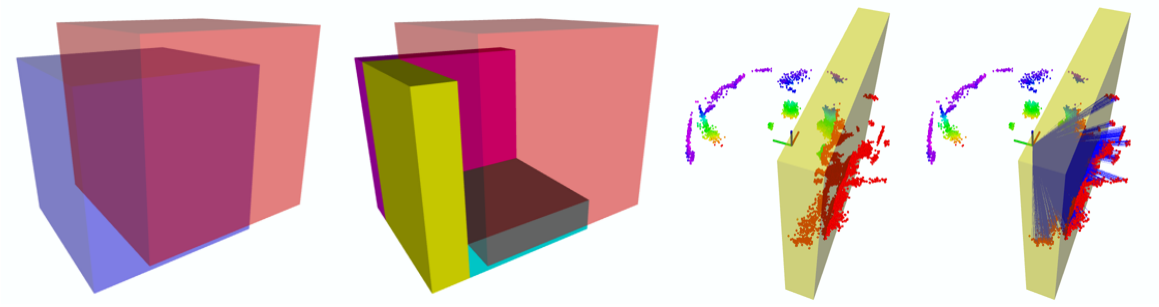
For points with norms less than a user-specified maximum range r_d, the EM approach is adapted from [176] to accept points that lie within a Mahalanobis distance of λ. Because Gaussians fall off quickly, points far away from a given density will have a small effect on the updated parameters for that density. By reducing the number of points, this decreases the computational cost of the EM calculation. Only points that have a value smaller than λ are considered (i.e., points larger than λ are discarded):

$$\lambda > \sqrt{(\boldsymbol{x}_n - \boldsymbol{\mu}_m^1)^T (\boldsymbol{\Lambda}_{\boldsymbol{m}}^1)^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_m^1)} \tag{3.3}$$

where the superscript 1 denotes the initialized values for the mean, covariance, and weight. This approach differs from our prior work Tabib et al. [179]; we utilize the approach in [176] as it yields greater frame-to-frame registration accuracy in practice. Frame-to-frame registration is not used in this work and is left as future work.

3.2.2 Free Space

To learn a free space distribution, points with norms that exceed the maximum range r_d are projected to r_d. The EM approach from Sect. 3.2.1 is used to decrease the computational cost of learning the distribution. To further decrease the cost, the free space points are split into windows in image space and GMMs consisting of n_f components are learned for each window. The windowing strategy is employed for learning distributions over free space points because it yields faster results and the distributions cannot be used for frame-to-frame registration. The number of windows and components per window is selected empirically. Fig. 3.1d illustrates the effect of the windowing using colored patches and Fig. 3.1e illustrates the result



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

Figure 3.2: Overview of the method by which occupancy is reconstructed. (a) The blue bounding box $\mathbf{B}_{t+1}$ is centered around $\mathbf{X}_{t+1}$ and red bounding box $\mathbf{B}_t$ is centered at $\mathbf{X}_t$. (b) illustrates the novel bounding boxes in solid magenta, teal, and yellow colors that represent the set difference $b_{t+1} \setminus b_t$. (c) Given a sensor origin shown as a triad, resampled pointcloud, and novel bounding box shown in yellow, each ray from an endpoint to the sensor origin is tested to determine if an intersection with the bounding box occurs. The endpoints of rays that intersect the bounding box are shown in red. (d) illustrates how the bounding box occupancy values are updated. Endpoints inside the yellow volume update cells with an occupied value. All other cells along the ray (shown in blue) are updated to be free.

of this windowing technique with black densities. Once the free space distributions are learned for each window the windowed distributions are merged into a single distribution.

Let $\mathcal{G}_i(\boldsymbol{x})$ be a GMM trained from N_i points in window i and let $\mathcal{G}_j(\boldsymbol{x})$ be a GMM trained from N_j points in window j, where $\sum_{w=1}^{W} N_w = N$ for sensor observation $\mathcal{Z}_t$ and W windows. $\mathcal{G}_j(\boldsymbol{x}) = \sum_{k=1}^{K} \tau_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\nu}_k, \boldsymbol{\Omega}_k)$ may be merged into $\mathcal{G}_i(\boldsymbol{x}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$ by concatenating the means, covariances, and weights. However, care must be taken when merging the weights as they must be renormalized to sum to 1 [166]. The weights are renormalized via Eqs. (3.4) and (3.5):

$$N^* = N_i + N_j \tag{3.4}$$

$$\boldsymbol{\pi}^* = \begin{bmatrix} \frac{N_i \pi_1}{N^*} & \dots & \frac{N_i \pi_m}{N^*} & \frac{N_j \tau_1}{N^*} & \dots & \frac{N_j \tau_k}{N^*} \end{bmatrix}^T \tag{3.5}$$

where $m \in [1, \dots, M]$ and $k \in [1, \dots, K]$ denote the mixture component in GMMs $\mathcal{G}_i(\boldsymbol{x})$ and $\mathcal{G}_j(\boldsymbol{x})$, respectively. $N^* \in \mathbb{R}$ is the sum of the support sizes of $\mathcal{G}_i(\boldsymbol{x})$ and $\mathcal{G}_j(\boldsymbol{x})$. $\boldsymbol{\pi}^* \in \mathbb{R}^{M+K}$ are the renormalized weights. The means and covariances are merged by concatenation.

3.2.3 Local Occupancy Grid Map

The occupancy grid map [184] is a probabilistic representation that discretizes 3D space into finitely many grid cells $\mathbf{m} = \{m_1, ..., m_{|\mathbf{m}|}\}$. Each cell is assumed to be independent and the probability of occupancy for an individual cell is denoted as $p(m_i|\mathbf{X}_{1:t}, \mathcal{Z}_{1:t})$, where $\mathbf{X}_{1:t}$ represents all vehicle states up to and including time t and $\mathcal{Z}_{1:t}$ represents the corresponding observations. Unobserved grid cells are assigned the

uniform prior of 0.5 and the occupancy value of the grid cell m_i at time t is expressed using log odds notation for numerical stability.

$$l_{t,i} \triangleq \log\left(\frac{p(m_i|\mathcal{Z}_{1:t}, \mathbf{X}_{1:t})}{1 - p(m_i|\mathcal{Z}_{1:t}, \mathbf{X}_{1:t})}\right) - l_0$$

When a new measurement $\mathcal{Z}_t$ is obtained, the occupancy value of cell m_i is updated as

$$l_{t,i} \triangleq l_{t-1,i} + L(m_i|\mathcal{Z}_t)$$

where $L(m_i|\mathcal{Z}_t)$ denotes the inverse sensor model of the robot and l_0 is the prior of occupancy [184].

Instead of storing the occupancy grid map $\mathbf{m}$ that represents occupancy for the entire environment viewed since the start of reconstruction onboard the vehicle, a local occupancy grid map $\bar{\mathbf{m}}_t$ is maintained centered around the robot's pose $\mathbf{X}_t$. The local occupancy grid map moves with the robot, so when regions of the environment are revisited, occupancy must be reconstructed from the surface models $\mathcal{G}(\boldsymbol{x})$ and $\mathcal{F}(\boldsymbol{x})$. To reconstruct occupancy at time $t+1$ given $\bar{\mathbf{m}}_t$, the set difference of the bounding boxes b_t and b_{t+1} for $\bar{\mathbf{m}}_t$ and $\mathbf{m}_{t+1}$, respectively, are used to compute at most three non-overlapping bounding boxes (see Figs. 3.2a and 3.2b for example). The intersection of the bounding boxes remains up-to-date, but the occupancy of the novel bounding boxes must be reconstructed using the surface models $\mathcal{G}(\boldsymbol{x})$ and $\mathcal{F}(\boldsymbol{x})$. Raytracing is an expensive operation [5], so time is saved by removing voxels at the intersection of b_t and b_{t+1} from consideration.

The local occupancy grid map at time $t+1$, $\bar{\mathbf{m}}_{t+1}$, is initialized by copying the voxels in local grid $\bar{\mathbf{m}}_t$ at the intersection of b_{t+1} and b_t. In practice, the time to copy the local occupancy grid map is very low (on the order of a few tens of milliseconds) as compared to the cost of raytracing through the grid. Not all Gaussian densities will affect the occupancy reconstruction so to identify the GMM components that intersect the bounding boxes a KDTree [16] stores the means of the densities. A radius equal to twice the sensor's max range is used to identify the components that could affect the occupancy value of the cells in the bounding box. A ray-bounding box intersection algorithm [192] checks for intersections between the bounding box and the ray from the sensor origin to the density mean. Densities that intersect the bounding box are extracted into local submaps $\bar{\mathcal{G}}(\boldsymbol{x})$ and $\bar{\mathcal{F}}(\boldsymbol{x})$. Points are sampled from each distribution and raytraced to their corresponding sensor origin to update the local grid map (example shown in Figs. 3.2c and 3.2d).

As the number of mixture components in the distribution increases over time in one region, updating the occupancy becomes increasingly expensive as the number of points needed to resample and raytrace increases. Tabib et al. [179] detail a method for limiting the potentially unbounded number of points from a 360° field-of-view (FoV) LiDAR sensor. An extension to limited FoV sensors is shown in [180].

CHAPTER 4

Motion Primitives-based Planning for Active Reconstruction

This chapter presents the motion primitives-based planner developed in this thesis to address the challenge **C1**. The action space used by the motion planner is represented as a collection of forward-arc motion primitives with velocity bounds derived from sensing and kinodynamic capabilities. Anytime planning (**R1**) is enabled using this action space through a receding-horizon action selection strategy that leverages Monte Carlo tree search (MCTS). Kinodynamic feasibility (**R2**) is ensured through a maximum acceleration and jerk checks along candidate motion primitives during MCTS.

Section 4.1 details this approach. The simulation experiments (Sect. 4.2.2) study the feasibility of the action selection approach for operation onboard an aerial robot (Sect. 4.2.1) and the efficacy of the action representation towards rapid reconstruction. The real-world experiments evaluate the rate of reconstruction and the maximum speeds achieved by the robots in environments with and without obstacles. A summary of this chapter is provided in Sect. 4.3.

4.1 Approach

This section presents an idealized analysis in Sect. 4.1.1 that is used in the motion primitive library design for the action representation (Sect. 4.1.2) of the receding-horizon reconstruction planner detailed in Sect. 4.1.3.

4.1.1 Steady-State Velocity Analysis

This section presents analysis of reconstruction performance for an aerial robot operating for steady-state conditions such as continuous motion toward a frontier. This analysis produces bounds on velocity and rates of entropy reduction, given constraints on dynamics and sensing. We leverage these insights in Sect. 4.1.2 to design motion primitive actions for rapid reconstruction.

System Model and Safety Constraints This work applies a simplified double-integrator quadrotor model with acceleration and velocity constraints for analysis of limits on reconstruction performance, which can be thought of as a relaxation

of dynamics models that are commonly used for position and attitude control of multirotor vehicles [126, 119]. Let $\mathbf{r} = [x, y, z]^{\mathrm{T}}$ be the position of the vehicle in an inertial world frame $\mathcal{W} = \{\mathbf{x}_{\mathcal{W}}, \mathbf{y}_{\mathcal{W}}, \mathbf{z}_{\mathcal{W}}\}$, and let the body frame be $\mathcal{B} = \{\mathbf{x}_{\mathcal{B}}, \mathbf{y}_{\mathcal{B}}, \mathbf{z}_{\mathcal{B}}\}$. Assuming small roll and pitch angles, and given the yaw angle ψ, the system state is $\boldsymbol{\xi} = [\mathbf{r}^{\mathrm{T}}, \psi, \dot{\mathbf{r}}^{\mathrm{T}}, \dot{\psi}]^{\mathrm{T}}$. The derivatives of position and yaw satisfy bounds on velocity and acceleration

$$||\dot{\mathbf{r}}||_2 \leq V_{\max} \qquad ||\ddot{\mathbf{r}}||_2 \leq A_{\max} \qquad |\dot{\psi}| \leq \Omega_{\max} \tag{4.1}$$

where $||\cdot||_2$ is the L2-norm.

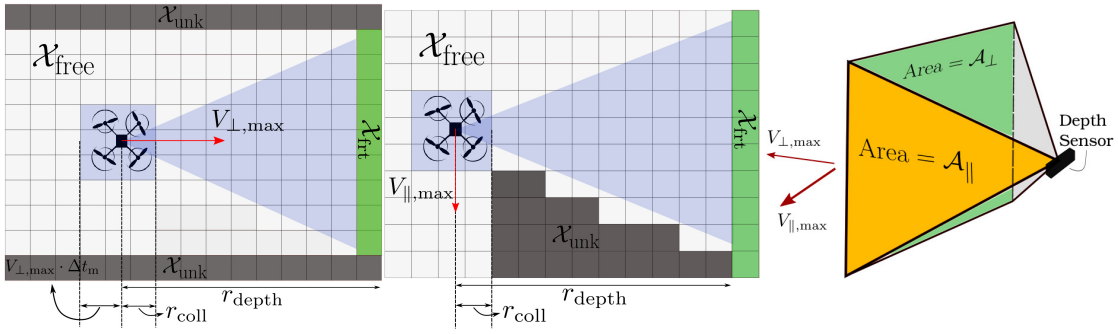
Further, the robot is equipped with a forward-facing depth sensor with range of $z_{\max}$ for use in mapping. However, while navigating the environment, the robot must also be able to avoid collisions with obstacles.

The requirement for collision-free operation restricts the set of actions that a multirotor can safely execute while navigating in an unknown environment. A planning policy can ensure collision-free operation by guaranteeing that the robot is able to stop entirely within unoccupied space $\mathcal{W}_{\text{free}}$, given an appropriate collision radius r_{coll}, such as in the work of Janson et al. [93]. In the worst case, any voxel in the unknown space $\mathcal{W}_{\text{unk}}$ may be revealed to be occupied and so possibly force the robot to stop within $\mathcal{W}_{\text{free}}$.

The robot plans once every Δt_{p} seconds, and there is also latency Δt_{m} for acquiring depth information and integrating it into the occupancy map. The sensor data is Δt_{m} seconds old at the beginning of planning, and once the planner is done, the robot executes the selected action for another Δt_{p} so that the total effective latency is no more than $\Delta t_{\text{l}} = \Delta t_{\text{m}} + 2\Delta t_{\text{p}}$. Note that, although latency may be unpredictable in practice, the robot will not depend on consistent latency to maintain safe operation.

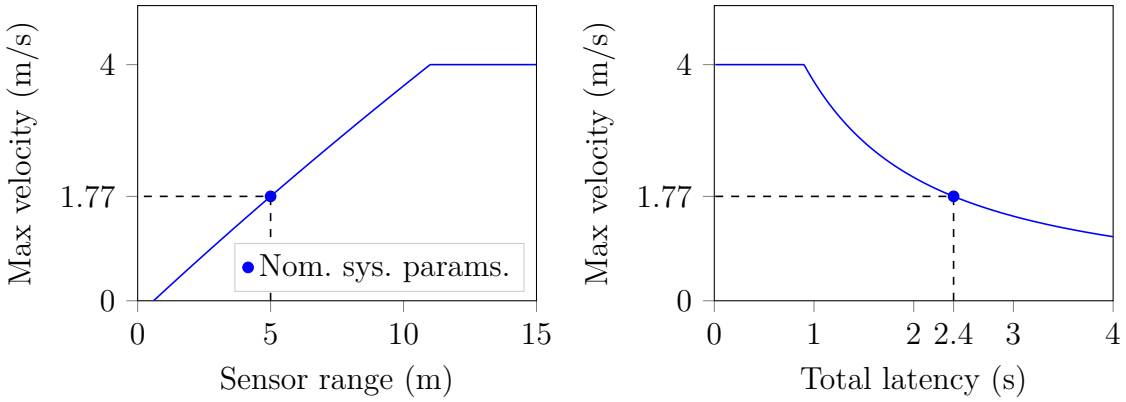
Steady-State reconstruction Scenarios Figure 4.1 illustrates two possible scenarios for steady-state motion with respect to a frontier. For the perpendicular case (Fig. 4.1a) the robot moves continuously toward a frontier and may have to avoid obstacles at the edge of the sensor range. As discussed in Sect. 4.1.1, the robot must be able to avoid collisions with obstacles in the unknown environment even if there are not any there. This means that the the robot must always be prepared to stop before reaching the frontier. For the parallel case (Fig. 4.1b) the robot moves along the frontier through space that has already been mapped. When known space is free of obstacles, the robot may continue to do so at the maximum allowable velocity. This scenario can also be thought of as the limit for a spiral motion—which will arise later in the experimental results—as the curvature becomes very small.

Bounds on Velocity Given the system model and constraints for reconstruction scenarios, we now proceed with calculation of the velocity bounds for motion perpendicular and parallel to the frontier, $V_{\perp,\max}$ and $V_{\parallel,\max}$ respectively. Maximum velocity toward the frontier is computed based on motion at a constant velocity followed by stopping at maximum deceleration to satisfy the safety constraint. The expression



(a) Perpendicular Scenario (b) Parallel Scenario (c) Sensor Cone

Figure 4.1: Steady-state reconstruction scenarios. Analysis in Sect. 4.1.1 presents upper bounds on velocities are for a double integrator system **(a)** for motion perpendicular to a frontier ($V_{\perp,\max}$) and **(b)** for motion parallel to it ($V_{\parallel,\max}$). To ensure safety in the perpendicular case, the robot has to stop within a user-specified collision radius from the frontier ($\mathcal{W}_{\text{frt}}$), i.e. within $z_{\max} - r_{\text{coll}}$ from the current state. For the parallel case, no such restrictions exist since the robot is moving in the explored space which, ideally, is free ($\mathcal{W}_{\text{free}}$). **(c)** Combining the area of the projection of the sensor cone in the direction of motion with the bounds on velocity leads to upper bounds on rates of novel voxels observed during reconstruction.



(a) Velocity ($V_{\perp,\max}$) vs. Sensor Range ($z_{\max}$) (b) Velocity ($V_{\perp,\max}$) vs. Total Latency (Δt_{l})

Figure 4.2: Maximum achievable velocity moving toward a frontier ($V_{\perp,\max}$) according to Eq. (4.2) based on parameters used in Table 4.1 and varying **(a)** sensor range and **(b)** total latency (which consists of latency for the mapping system and time for planning). The circle marks the maximum velocity at which the robot can move toward unknown space for the parameters used in this chapter (see Table 4.1) which is less than half the overall velocity bound. Approaching this velocity limit requires either sensor range exceeding 10 m, both decreased planning time and mapping latency, or some combination of the two.

Value/Cases	Area (m^2)	Velocity (m/s)	Volume rate (m^3/s)	Entropy rate (bits/s)
Perpendicular ($\perp$)	56.4	1.77	99.83	1.25×10^4
Parallel ($\parallel$)	57.19	4.00	228.8	2.86×10^4
$\perp$, rapid yaw	56.8	1.77	100.5	1.26×10^4
$\parallel$, rapid yaw	78.05	4.00	312.2	3.90×10^4

Table 4.1: Steady-state upper bounds on velocity and rate of entropy reduction for the scenarios described in Sect. 4.1.1. All values are computed for a planning time of 1 Hz, mapping latency 0.4 s, sensor point cloud of size 9.93 m × 5.68 m based on the RealSense D435 depth sensor with image size 424px × 240px and a maximum depth $z_{\max}$ of 5 m. Occupancy grid resolution is 20 cm with an overall bound on top speed $V_{\max}$ at 4 m/s, and collision radius r_{coll} set at 0.6 m.

for $V_{\perp,\max}$ is a function of acceleration ($A_{\max}$), maximum sensing range ($z_{\max}$), the collision radius (r_{coll}), and the latency in planning Δt_{l} (see Fig. 4.1a) and is given by

$$\begin{aligned} V_{\perp,\max} &= \min(V_{\max}, V'_{\perp,\max}) \\ V'_{\perp,\max} &= A_{\max} \cdot \left(\sqrt{\Delta t_{\text{l}}^2 + 2\frac{z_{\max} - r_{\text{coll}}}{A_{\max}}} - \Delta t_{\text{l}} \right). \end{aligned} \tag{4.2}$$

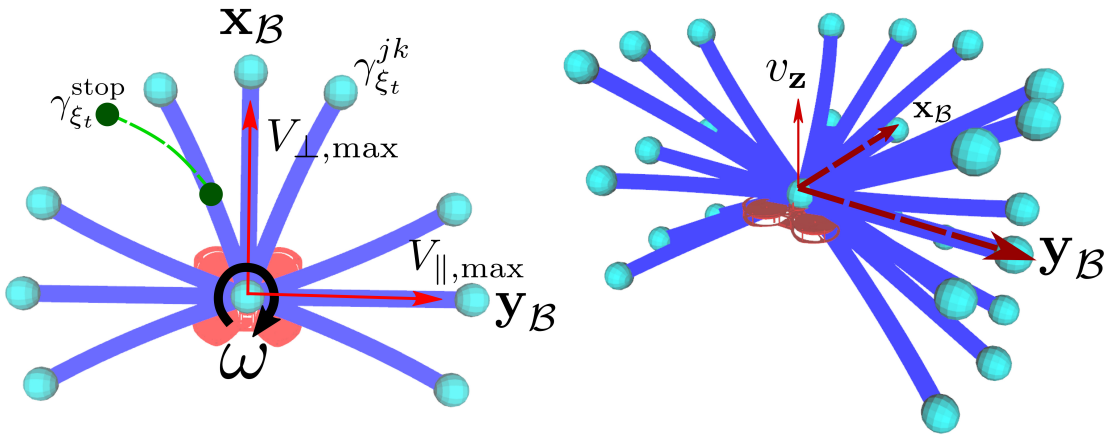
Figure 4.2 shows the variation of this bound with $z_{\max}$ and Δt_{l} for the parameters used in this chapter. For motion parallel to a frontier (see Sect. 4.1.1 and Fig. 4.1b), there are no obstacles in the direction of motion. Therefore, the steady-state upper bound on the velocity moving parallel to the frontier is identical to the maximum achievable by the system, i.e. $V_{\parallel,\max} = V_{\max}$.

The entropy reduction then can also be bounded for each scenario terms of the sensor geometry (see Fig. 4.1c) and steady-state velocities by projecting the sensing volume in the direction of motion. Here, we also introduce the possibility of rapid yaw motion during either motion. Results are shown in Table 4.1. Note that moving parallel to the frontier can provide significantly improved performance.

4.1.2 Action Representation

This section details the design of available actions for the proposed motion planning framework. We define a trajectory generation scheme, related parameters and conventions, and action design specifics leveraging insights gained in Sect. 4.1.1.

Motion Primitive Library Generation Safe and accurate high-speed flight requires large and smooth linear acceleration and angular velocity references. Smoothness for such references depends on higher derivatives of position, jerk and snap [125]. For this work, the actions that are available to the robot are snap-continuous, forward arc [203] motion primitives, which have previously been applied to high-speed teleoperation of multirotors [165]. Given the differentially-flat state of the multirotor at time t, $\boldsymbol{\xi}_t = [x, y, z, \psi]^{\mathrm{T}}$, denote an available action parameterization as $\mathbf{a} = [v_{\mathbf{x}}, v_{\mathbf{z}}, \omega]$ where $v_{\mathbf{x}}$ and $v_{\mathbf{z}}$ are velocities in the body frame $\mathbf{x}_{\mathcal{B}}$ and $\mathbf{z}_{\mathcal{B}}$ directions, and ω is the



(a) Variation in ω (b) Variation in $v_\mathbf{z}$

Figure 4.3: Actions in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane at the multirotor state $\boldsymbol{\xi}_t$, $\gamma_{\boldsymbol{\xi}_t}^{jk}$ (blue), are generated using discretized velocity bounds obtained from the analysis in Sect. 4.1.1. The set of such primitives at each state is termed a motion primitive library (MPL) $\Gamma_{\boldsymbol{\xi}_t}$. MPLs are sampled in directions perpendicular ($\mathbf{x}_\mathcal{B}$) and parallel ($\mathbf{y}_\mathcal{B}, -\mathbf{y}_\mathcal{B}$) to the sensor scans with speeds bounded by $V_{\perp,\max}$ and $V_{\parallel,\max}$ respectively, see **(a)**. Variation in $\mathbf{z}_\mathcal{B}$ direction using a user-specified bound on vertical velocity, $V_\mathbf{z}$, yields the final action space shown in **(b)**. Dynamically feasible stopping trajectories $\gamma_{\boldsymbol{\xi}_t}^{\text{stop}}$ are available for each primitive (green) for safety (only one shown for brevity).

body yaw rate. Actions are discretized using user-specified maximum velocity bounds in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane (ω variation, N_ω primitives) and in $\mathbf{z}_\mathcal{B}$ direction ($v_\mathbf{z}$ variation, $N_\mathbf{z}$ primitives) to obtain a motion primitive library (MPL) $\Gamma_{\boldsymbol{\xi}_t}$ given by (Fig. 4.3):

$$\Gamma_{\boldsymbol{\xi}_t} = \{\gamma_{\boldsymbol{\xi}_t}^{jk} \mid j \in [1, N_\omega], k \in [1, N_\mathbf{z}], \|[v_\mathbf{x}, v_\mathbf{y}]\| \leq V_{\max}, \|v_\mathbf{z}\| \leq V_\mathbf{z}, \|\omega\| \leq \Omega_{\max}\} \quad (4.3)$$

where, $\|\cdot\|$ denotes L2-norm of a vector, $V_{\max}$ and $V_\mathbf{z}$ are the bounds on speed in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane and $\mathbf{z}$ direction respectively, and $\Omega_{\max}$ is the bound on body yaw rate. For a given action discretization, the motion primitive $\gamma_{\boldsymbol{\xi}_t}^{jk}$ is an 8^{th} order polynomial in time with fixed start and end point velocities and unconstrained position at the end. The velocity at the end point, at time τ, follows by forward propagating unicycle kinematics using the current state and the action parameterization while higher order derivatives up to snap, endpoints are kept zero:

$$\dot{\boldsymbol{\xi}}_\tau = [v_\mathbf{x} \cos\psi, v_\mathbf{x} \sin\psi, v_\mathbf{z}, \omega]^\mathrm{T}, \ \psi = \omega\tau, \ \boldsymbol{\xi}_\tau^{(j)} = \mathbf{0}, \text{ for } j \in \{2, 3, 4\} \quad (4.4)$$

where $\boldsymbol{\xi}^{(j)}$ denotes the j^{th} time derivative of $\boldsymbol{\xi}$. The stopping trajectories at any $\boldsymbol{\xi}_t$ ($\gamma_{\boldsymbol{\xi}_t}^{\text{stop}}$, Fig. 4.3) can be sampled by keeping $\dot{\boldsymbol{\xi}}_t = \mathbf{0}$.

In contrast to the fixed duration (τ) primitives presented in [165], the duration for the primitive is kept minimum via a line search from the minimum possible duration (Δt_p) to the user-specified maximum duration ($\tau_{\max}$). The search terminates when the first dynamically feasible motion primitive is obtained. This dynamic feasibility check is based on pre-specified empirically observed bounds on linear acceleration and

linear jerk $L2$-norms. This search achieves having the trajectory in the desired end point velocity $\dot{\boldsymbol{\xi}}_t$ in the minimum time possible from the current state.

Action Space for Fast reconstruction The action space for the proposed planner is a collection of MPLs, defined by Eq. (4.3), generated using linear velocities based on bounds obtained in Sect. 4.1.1, $V_{\perp,\max}$ and $V_{\parallel,\max}$. The planner uses 6 MPLs to represent the action space, $\mathcal{A}_{\text{act}} = \{\Gamma^i_{\boldsymbol{\xi}_t} \mid i \in [1,6]\}$, and sets of upper bounds on linear velocities (Table 4.2) define each of these different MPLs. These MPLs include both high-speed actions for navigating the environment and actions that mimic steady-state conditions described Sect. 4.1.1. Later, in Sect. 4.2, we highlight effects on reconstruction performance due to these components, especially the high speed parallel and low speed perpendicular MPLs.

4.1.3 Action Selection

We formulate the action selection problem as a receding-horizon optimization seeking to maximize cumulative information gain [30], and build upon previous work [106, 41, 43] on robotic reconstruction using Monte Carlo tree search (MCTS). The receding-horizon formulation allows anytime behavior (**R1**).

Most MCTS-based planners follow four steps: selection, expansion, simulation playout, and backpropagation of statistics [31, 21]. Such planners usually construct a search tree iteratively by random rollout from a previously unexpanded node selected based on upper-confidence bounds for trees (UCT) [21]. Prior works [41, 43] have applied MCTS in planning for reconstruction using multirotors using a UCT-based selection policy, information gain rewards, and random simulation playout over a finite horizon. We extend this approach by adding considerations for model constraints into the node expansion phase of MCTS.

Information-Theoretic reconstruction Objectives Following a similar approach as our prior work [43], the planner optimizes an objective with two components: a local information reward based on Cauchy-Schwarz quadratic mutual information (CSQMI) [30], and a global reward for decrease the shortest path distance to points in the state space that are expected to provide significant information gain [43]. For any candidate action, $\gamma_{\boldsymbol{\xi}_t}$, we compute the local information gain $\mathcal{I}_\gamma$ over user-specified time intervals and treat the joint information gain as a reward for the MCTS planner. The distance reward serves to direct the robot toward unexplored and possibly distant regions of the environment once the local environment is exhausted of information causing the local information reward to decrease. Alternatively, designers might substitute competing routing and scheduling approaches [103] for the distance cost subject to with tradeoffs in computational cost and system design.

Safety at High Speeds Given the action representation described in Sect. 4.1.2, we require the planner to ensure safety while expanding nodes. Specifically, the trajectory tracked by the controller should both respect constraints on the dynamics and remain

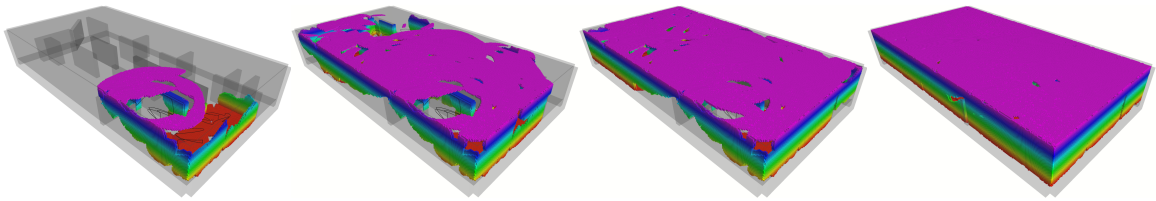
Type	**Max. Speed**	Dir.	N_ω	$N_{\mathbf{z}}$	N_{prim}
Yaw	0	ψ	1	1	1
$\perp$	$V_{\perp,\max}$	$\mathbf{x}_\mathcal{B}$	9	5	45
$\perp$	$V_{\max}$	$\mathbf{x}_\mathcal{B}$	9	5	45
$\parallel$	$V_{\max}$	$\mathbf{y}_\mathcal{B}$	9	5	45
$\parallel$	$V_{\max}$	$-\mathbf{y}_\mathcal{B}$	9	5	45
Z	$V_{\mathbf{z}}$	$\mathbf{z}$	1	5	5

(a) Large Library

Type	**Max. Speed**	Dir.	N_ω	$N_{\mathbf{z}}$	N_{prim}
Yaw	0	ψ	1	1	1
$\perp$	$V_{\perp,\max}$	$\mathbf{x}_\mathcal{B}$	3	3	9
$\perp$	$V_{\max}$	$\mathbf{x}_\mathcal{B}$	3	3	9
$\parallel$	$V_{\max}$	$\mathbf{y}_\mathcal{B}$	3	3	9
$\parallel$	$V_{\max}$	$-\mathbf{y}_\mathcal{B}$	3	3	9
Z	$V_{\mathbf{z}}$	$\mathbf{z}$	1	3	3

(b) Minimal Library

Table 4.2: Motion primitive libraries used to construct action space using Eq. 4.3 from Sect. 4.1.2 and the bounds obtained after applying analysis presented in Sect. 4.1.1. The vertical velocity bound ($V_{\mathbf{z}}$) and the speed bound in $\mathbf{x}_\mathcal{B}-\mathbf{y}_\mathcal{B}$ plane ($V_{\max}$) are kept at 0.3 m/s and 4.0 m/s respectively. The total number of primitives for a MPL is $N_{\text{prim}} = N_\omega \cdot N_{\mathbf{z}}$.



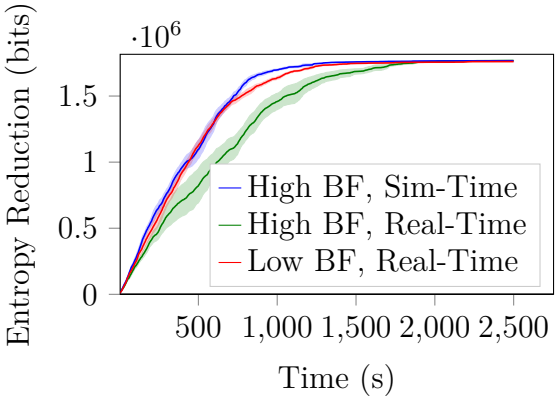
(a) $t = 100\,\text{s}$ (b) $t = 500\,\text{s}$ (c) $t = 1000\,\text{s}$ (d) $t = 1500\,\text{s}$

Figure 4.4: Occupied map at different stages of reconstruction of a simulated three-dimensional 60 m × 30 m × 11 m warehouse environment used for experiments. The map is colored based on Z height.

in known free space for all time. To satisfy this condition (**R2** and **R3**), before sending any trajectory to the robot, we require knowledge of a trajectory that will bring the robot to a stop—and potentially keep it there—afterward. As such, the robot will avoid collision, even if the planner fails to provide any output. If ever planning fails, the known stopping trajectory is sent to the robot, and the robot will continue to replan from a future reference point. Collision checking is performed at sampled points along the motion primitive associated with the node through a signed distance field obtained using the occupancy information in the map representation. The kinodynamic feasibility of a node is checked through user-specified maximum acceleration and jerk constraints over sampled points along the associated motion primitive.

4.2 Results

This section describes hardware and simulation results for the proposed reconstruction approach. The simulation results evaluate performance in a warehouse-like environment which serves as a representative example of a large-scale reconstruction task. The hardware results demonstrate reconstruction at high speeds using a hexarotor platform under various degrees of clutter. Unless otherwise noted, the configuration



Time to reach (s):	**25%**	**50%**	**75%**	**90%**
High BF, Sim-Time	**157.7**	**365.7**	**602.5**	**788.3**
High BF, Real-Time	267.4	501.6	866.5	1183.2
Low BF, Real-Time	193.2	382.5	604.8	901.1

Figure 4.5: (Top) Entropy reduction vs. time for each of the simulation trials. Transparent patches show standard-error. (Bottom) Average time to reach fractions of the maximum entropy reduction over all trials (1.766×10^6 bits). While the High BF, Sim-Time case dominates in terms of entropy reduction, the Low BF, Real-Time case is able to provide similar performance. Note that the different configuration are described at the beginning of Sect. 4.2.2 and that BF denotes branching factor.

Actions:	$\perp$	$\parallel$	**Z**	**Yaw**	**Stop**
Selection frequency	0.343	**0.479**	0.089	0.088	0.001
Total entropy red, norm.	**0.40**	**0.41**	0.06	0.12	0.01
Average speed (m/s)	2.163	**2.778**	0.959	1.114	1.611
Average yaw rate (rad/s)	0.286	0.234	0.170	**0.381**	0.080
Entropy red. rate (bits/s):	**2425**	**2389**	2020	**2414**	1283

Table 4.3: Performance statistics for the high branching factor, Sim-Time simulation study. Unless otherwise stated, rates refer to average performance over time-periods when tracking a given action type. All statistics include any time the robot is tracking a given action, except for the entropy reduction rate in the last row, which is conditional on a significant entropy reduction rate (greater than 600 bits/s). Best (or nearly equivalent) values are bolded.

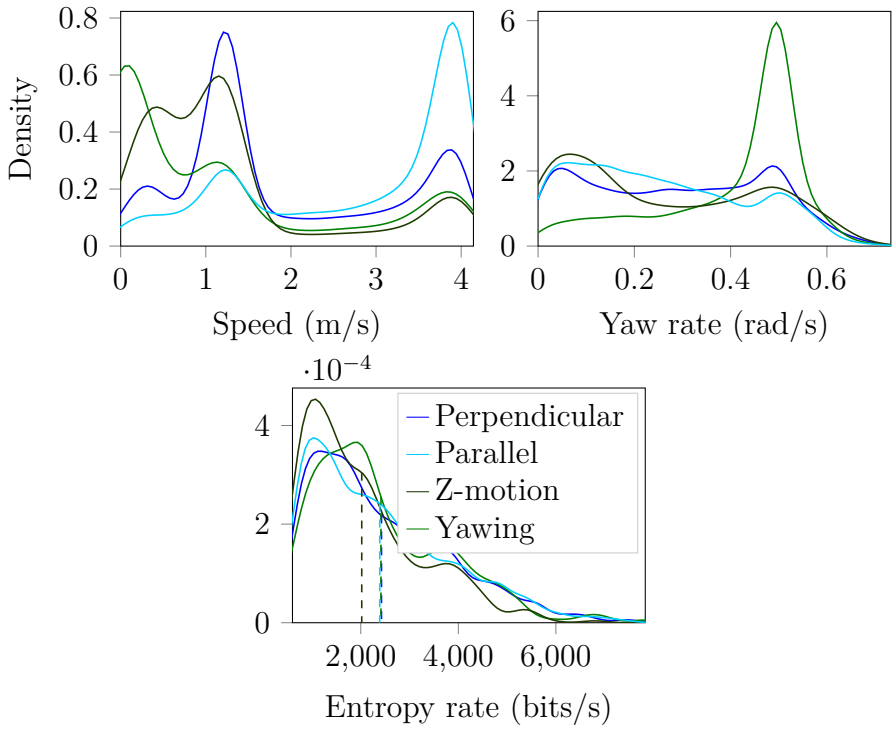


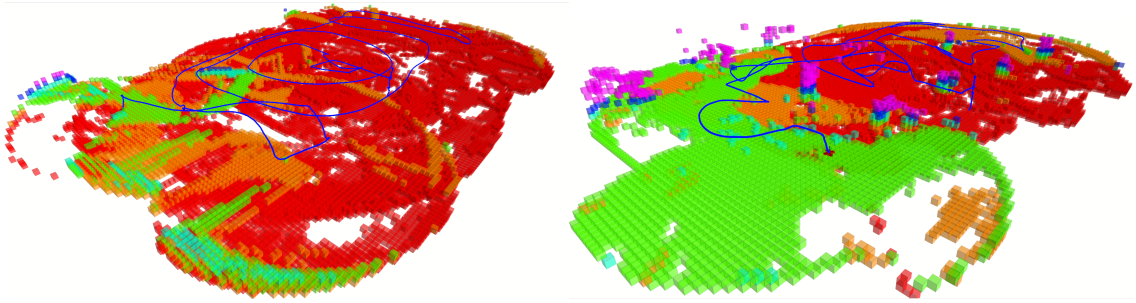
Figure 4.6: reconstruction performance by action. The plots provide estimates of probability densities (via kernel density estimation) for speed, yaw rate, and entropy rate conditional on the type of action (Table 4.2) being executed by the robot. All densities are also conditional on a significant entropy reduction rate (greater than 600 bits/s) in order to emphasize performance characteristics for actions that directly contribute to the map rather than traversal of known space or time periods after reconstruction is effectively complete. Note that, even though the planner has access to high-speed motions perpendicular to frontiers, entropy reduction for perpendicular actions occurs primarily at lower speeds (1.25m/s) in accordance with the analysis in Sect. 4.1.1.

of the robot for simulation matches the hardware.

4.2.1 Aerial Platform

The platform used for experiments is a hexarotor aerial robot (Fig. 4.7a) with a forward-facing depth camera for mapping (Realsense D435) with a 89.57° by 59.24° field of view and a range of $z_{\max} = 5.0\,\text{m}$. The robot itself weighs 55.37 N, has a thrust to weight ratio of 3.5, and has a diameter of 0.89 m from motor to motor. Limits on acceleration and jerk are set to $A_{\max} = 10\,\text{m/s}^2$ and $J_{\max} = 35\,\text{m/s}^3$ respectively, based on empirical data available for the platform. Unless otherwise stated, the planning horizon is kept at 4 seconds for all experiments. The map is represented as a dense voxel grid that creates a consistent representation of the environment using the depth data. For both simulation and hardware experiments, mapping and planning run on a computationally constrained quad-core embedded CPU Gigabyte Brix 6500U. The robot obtains odometry estimates via a downward-facing camera and IMU using a

(a) Multirotor with Depth Sensor (b) Open Space Scenario (c) Obstacles Scenario



(d) Open space scenario reconstruction after 90 s (e) Obstacles scenario reconstruction after 90 s

Figure 4.7: **(a)** Multirotor used for hardware experiments in two real world scenarios: **(b)** open space and **(c)** space with scattered obstacles. **(d)** and **(e)** show the explored maps (color gradient based on Z height) and the overall paths of the robot after 90 s of 3D reconstruction using the proposed reconstruction approach.[1]

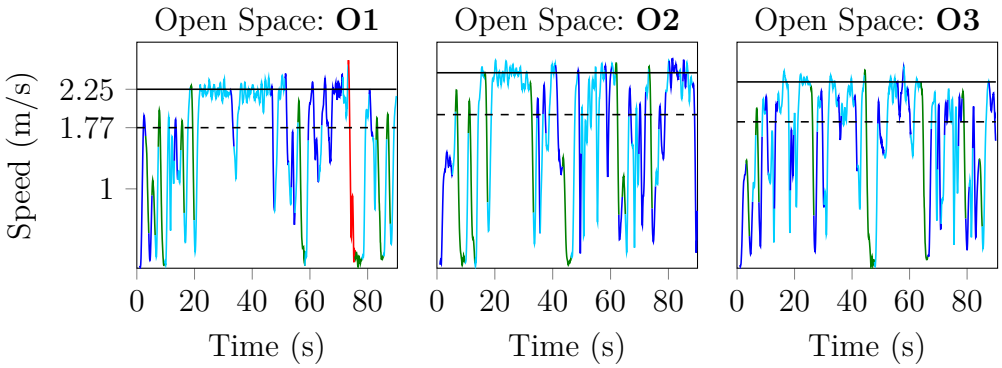
[1] Video: https://youtu.be/YXt4yiTpOAc

monocular Visual-Inertial Navigation System (VINS-Mono [148]), previously used for high-speed teleoperation of a multirotor [165]. This state estimation component, only present for hardware experiments, is executed on a quad-core NVIDIA Tegra TX2 on-board the vehicle. Contrary to perfect state estimation in simulation experiments, for the hardware experiments the robot only has access to odometry for navigation and is susceptible to drift. For the purpose of this work, we will continue to emphasize the role of planning and speed in the reconstruction experiments and will comment briefly on ramifications of drift on outcomes. Future iterations of this platform will seek to combine a local mapping strategy [43] with a complete SLAM system.

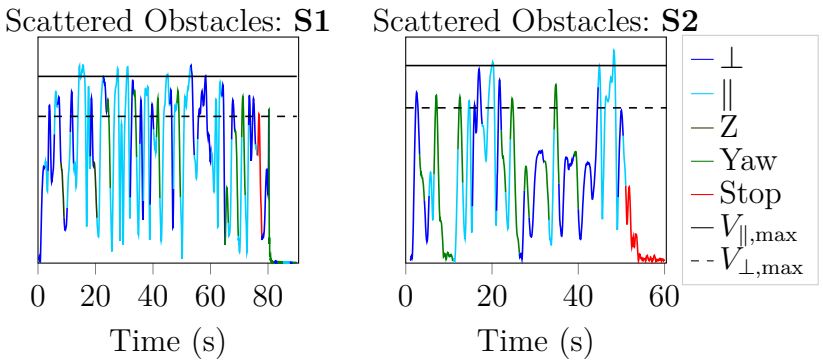
4.2.2 Simulated reconstruction of a Warehouse Environment

The simulations demonstrate reconstruction of a large warehouse environment (pictured in Fig. 4.4). These trials are repeated for three system configurations which vary the motion primitive library and the computational setting:

- **High branching factor (BF), Sim-Time**: The planner uses the large motion primitive library (Table 4.2a) for reconstruction. The simulation and clock pause



(a) Speeds attained during open space scenario



(b) Speeds attained during obstacles scenario

Figure 4.8: **(a)** and **(b)** show speeds attained by the robot in the outdoor open space and obstacles scenario respectively.

at the end of each planning round until the MCTS planner completes a user-defined number (3000) of iterations. The simulation time then does not include this additional time spent in planning.

- **High BF, Real-Time**: The planner uses the large motion primitive library for reconstruction, but the simulation of the multirotor runs in real time. The planner runs in an anytime fashion on a computer comparable to the on-board computer used for flight experiments presented in Sect. 4.2.3 while simulators for the camera and dynamics run on a separate computer.

- **Low BF, Real-Time**: The planner uses the minimal motion library (Table 4.2b) for reconstruction and the computational configuration is the same as the above.

These experiments first establish baseline performance (High BF, Sim-Time) given access to a variety of motion primitives and a relatively large amount of planning time. The latter two configurations demonstrate online planning in a configuration that closely matches the hexarotor platform used in this chapter. These experiments seek to demonstrate the role of computational constraints in design of the motion

primitive library. For each configuration, we provide several trials, one for each of 5 given starting locations.

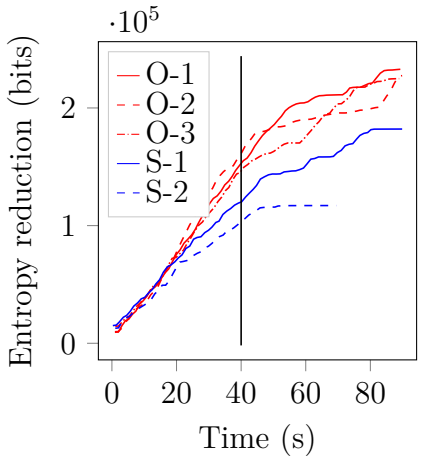
Each trial lasts 2500 seconds which provides ample time to explore the entire environment. For all trials, the perpendicular velocity $V_{\perp,\max}$ is further limited to 1.25 m/s, below the value of 1.77 m/s obtained in Sect. 4.1.1 which we find admits forward motion at a constant speed given the trajectory generation approach used for motion primitive design. The maximum speed is set to more than three times greater at $V_{\max} = 4.0$ m/s.

Figure 4.5 summarizes reconstruction performance for each experiment. The high branching factory Sim-Time case which has access to extra planning time performs at least as well or better than the other configurations in terms of entropy reduction. However, the configuration with same motion primitive library and real time is significantly impaired and requires between approximately 1.3 to 1.8 times as long to reach reported levels of entropy reduction. The lower branching factor case matches the first configuration much more closely. As such, this latter configuration is appropriate for deployment on the compute-constrained hexarotor platform.

In addition to being able to explore an environment rapidly and completely, we characterize the roles of the motion primitive actions in the reconstruction task. Figure 4.6 shows density estimates plots for speeds, yaw rates, and entropy rate labelled by the type of action selected by the MCTS planner for execution for periods when the entropy reduction rate is significant (greater than 600 bits/s) so as to separate reconstruction actions from traversal of the environment and time periods after reconstruction is effectively complete. This threshold corresponds to a knee point in the overall distribution of entropy reduction rates: 94.4% of all entropy reduction occurs above this threshold but during only 27.9% of time during the trials. Interestingly, the time rate of entropy reduction is largely consistent across action types. However, as expected, motions perpendicular to the frontier primarily contribute to entropy reduction at reduced speed despite both low-speed and high-speed primitives being available. Table 4.3 shows provides further statistics for the different kinds of actions. Even though entropy reduction rates are similar across actions when the entropy reduction rate is significant, the planner selects motions parallel and perpendicular to frontiers most frequently, and those actions account for more than 80% of all entropy reduction.

4.2.3 Hardware Experiments under Varied Conditions

Real-world autonomous reconstruction experiments are conducted using the aerial platform described in Sect. 4.2.1 (Fig. 4.7a) in two outdoor scenarios: (1) Open space (Fig. 4.7b), and (2) Scattered obstacles (Fig. 4.7c). The total reconstruction duration is limited to 90 seconds to minimize the effects of state estimation drift on the resulting map. During each scenario, the robots explores while confined to 12 m × 24 m × 2 m bounding box. The robot starts at the same position in the bounding box for each trial in both scenarios. The bounds on the speed for perpendicular ($V_{\perp,\max}$) and parallel ($V_{\parallel,\max}$) motions, are set at 1.25 m/s and 2.25 m/s respectively. The explored maps and robot trajectory for two experiments, one from each scenario, are shown in Figs. 4.7d



Trials:	**O1**	**O2**	**O3**	**S1**	**S2**
Entropy red. at 40s (bits$\times 10^5$)	1.53	1.61	1.46	1.2	1.04
Entropy red. final (bits$\times 10^5$)	2.32	2.27	2.25	1.82	1.16
Average speed (m/s)	1.50	1.56	1.56	1.42	0.98
Average yaw rate (rad/s)	0.39	0.39	0.33	0.33	0.31
Max. speed (m/s)	2.39	2.38	2.38	2.40	2.29
Max. yaw rate (rad/s)	0.66	0.59	0.55	0.60	0.63

Figure 4.9: Entropy reduction for hardware trials and summary statistics. Except for the final entropy reduction, all statistics are computed over the first 40 second of each trial (shown by the black bar in the entropy reduction plot).

and 4.7e. Speeds achieved by the vehicle during the experiments are shown in Fig. 4.8. Figure 4.9 provides plots of reduction of map entropy as well as summary statistics. Even though the trials were relatively short, the odometry often drifted significantly by the end This drift likely contributed to the robot getting stuck behind an obstacle during the trial **S2**. For this reason, we only use the first for the first 40 seconds of each trial when computing summary statistics unless otherwise noted.

As shown in Fig. 4.8, the odometry system reports that the robot reaches and slightly exceeds the maximum desired reference speed[2] in each trial, primarily while executing motions parallel to the frontier. Figure 4.7d shows a particularly notable example of this behavior where the robot executed an outward spiraling motion soon after the start of the trial.

4.3 Summary

In this chapter, a motion primitives-based planner for reconstruction is proposed that enables anytime operation (**R1**) while ensuring kinodynamically-feasible (**R2**) operation. The requirement **R1** is addressed through a receding-horizon action selection strategy based on MCTS where the reward function is the computationally

[2]The robot may exceed reference speeds due to error in tracking the position reference because of environment disturbances and inaccuracies in the system model.

expensive CSQMI function. The requirement **R2** is addressed with a modification of the expansion step in MCTS. Nodes are only expanded during search if they are kinodynamically-feasible, for both candidate actions and the associated stopping trajectories.

Simulation and real-world experiments are conducted to study the rapid reconstruction capabilities of the approach obtained after addressing these requirements (**C1**). The simulation experiments provide a motion primitive library configuration useful for computationally constrained operation onboard an aerial robot. Further, the utility of the action representation is characterized by studying the effect of each type of motion primitives on the reconstruction performance. The real-world experiments ascertain the deployability of this approach for rapid reconstruction purposes–the aerial robot achieves speeds up to 2.4 m/s in obstacle-rich environments.

CHAPTER 5
Adaptive-Speed Motion Primitives-based Teleoperation

This chapter details an adaptive-speed motion primitives-based teleoperation strategy towards addressing the challenge **C2** that requires developing adaptive-speed reconstruction. For the teleoperation task, modulation in the motion plan profiles (**R3**) is achieved via a hierarchical collision avoidance methodology that adapts the map resolution and utilizes the velocity bounds calculated in Chapter 4 to enable speed modulation.

Section 5.1 details this approach. The speed modulation capability is validated through simulated and real-world multirotor teleoperation experiments (Sect. 5.2) in a challenging cave environment (Fig. 1.1). A summary of this chapter is provided in Sect. 5.3.

5.1 Approach

This section details the adaptive-speed teleoperation approach. Figure 5.1 illustrates the components of the approach and how information flows between components. A local occupancy map is generated using depth observations, robot pose and voxel size (Sect. 5.1.1). Note that due to the myopic nature of the teleoperation task, only a few recent observations are required to be included in the local occupancy map. The motion primitive design block takes input from the operator's joystick and the voxel size of the local occupancy map. This information is used to compute a candidate motion primitive close to the operator's input and a fallback stopping motion primitive to use in case the next planning round fails (Sect. 5.1.2). The hierarchical collision avoidance block takes as input these two motion primitives and the local map to perform collision checking (Sect. 5.1.3). If both selected and stopping motion primitives are found to be feasible, they are sent to the controller for execution and the voxel size is increased for the next planning round. In case either of these primitives is infeasible, then the voxel size is decreased (and therefore the occupancy grid extents, see Fig. 5.2) and the process is repeated for the same planning round. If a suitable plan is not found, the stopping action from the previous planning round is executed.

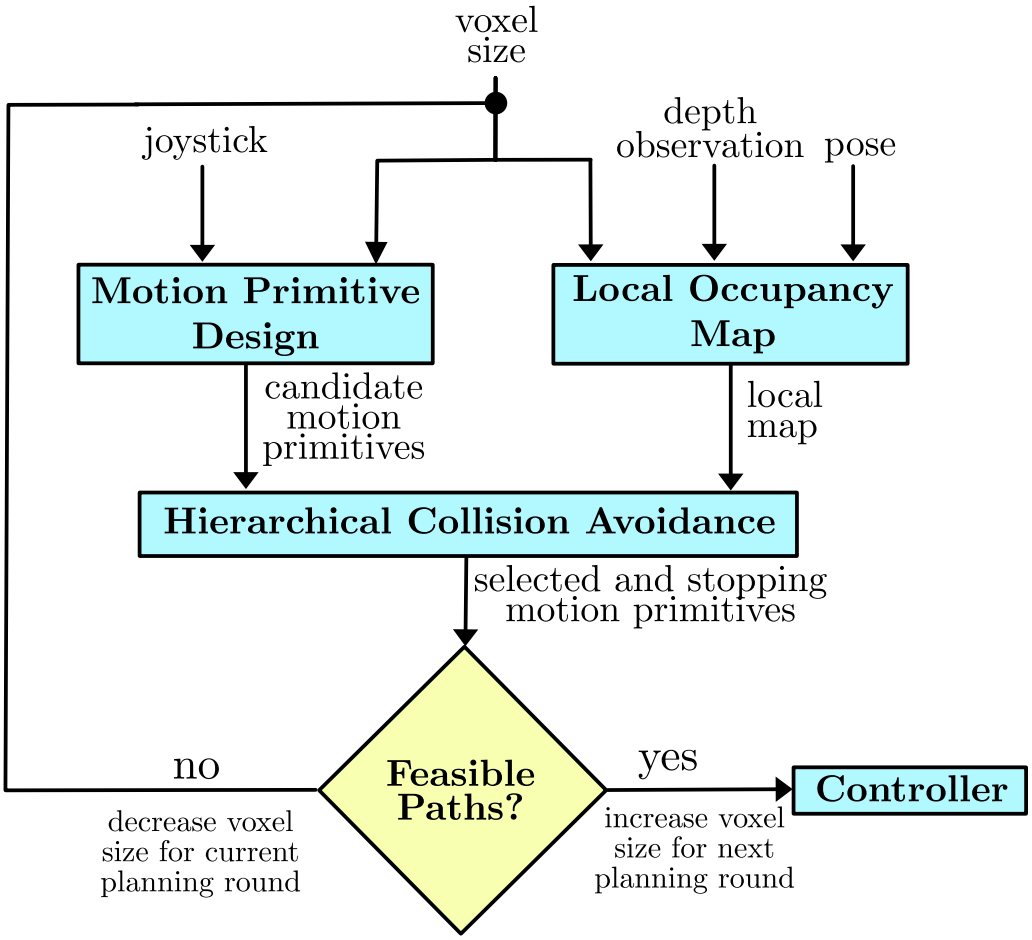
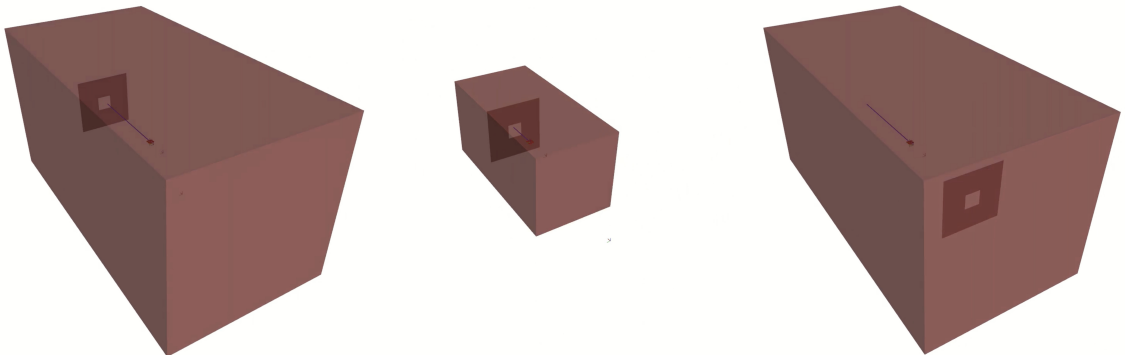


Figure 5.1: Information flow diagram for the technical approach.

5.1.1 Variable Resolution Local Occupancy Mapping

A three-dimensional local occupancy grid map $\mathbf{m}$ is generated in the body frame $\mathcal{B} = \{x_\mathcal{B}, y_\mathcal{B}, z_\mathcal{B}\}$ of the robot using the latest control state $\boldsymbol{\xi}^c$ as the origin, voxel size α, and number of voxels $|\mathbf{m}| = N_x \times N_y \times N_z$. The number of voxels along each dimension, $\{N_x, N_y, N_z\}$, are fixed throughout teleoperation but the voxel size α may vary during or across planning rounds. The local occupancy grid bounding box $\mathbf{B} = \{\mathbf{b}_{\min}, \mathbf{b}_{\max}\}$ extents are adjusted as a function of the voxel size, α. $\mathbf{b}_{\min} = \{x_{\min}, y_{\min}, z_{\min}\}$ is the minimum $x_\mathcal{B} - y_\mathcal{B} - z_\mathcal{B}$ coordinate of the bounding box and $\mathbf{b}_{\max} = \{x_{\max}, y_{\max}, z_{\max}\}$ is the maximum $x_\mathcal{B} - y_\mathcal{B} - z_\mathcal{B}$ coordinate of $\mathbf{B}$. Each voxel $m_i \in \mathbf{m}$ is a Bernoulli random variable whose value is 0 if it is free and 1 if it is occupied. Initially, the map is set to have a uniform occupancy probability in all voxels, $p(m_i) = 0.5, \forall\, i \in \{1, \ldots, |\mathbf{m}|\}$.

It is assumed that the multirotor is equipped with a limited field-of-view (FoV) forward-facing depth camera that provides a dense depth measurement $\mathcal{Z}$ at a user-specified sensing rate $1/\Delta t_\text{s}$. In addition to the latest state-measurement (keyframe), $\mathcal{K}_l = \{\boldsymbol{\xi}_l^c, \mathcal{Z}_l\}$, one past keyframe is maintained $\mathcal{K}_p = \{\boldsymbol{\xi}_p^c, \mathcal{Z}_p\}$. The past keyframe is selected based on a Euclidean distance threshold β between the latest state $\boldsymbol{\xi}_l^c$ and the previous keyframe state $\boldsymbol{\xi}_p^c$. Measurement $\mathcal{Z}_p$ is transformed into the frame



(a) (b) (c)

Figure 5.2: Bounding box extents for a scenario where the robot traverses a window. The teleoperator gives maximum joystick input in the forward direction for these three figures. (a) When the robot is far from the window, the bounding box extents and local occupancy map are large because the voxel size is also large. (b) As the multirotor gets closer to the window, the voxel size decreases and so does the bounding box extent because the number of voxels in the map stays the same. (c) After exiting the window, the bounding box expands to the original size. Note that the change in bounding box extents is achieved by varying the voxel size and keeping the number of voxels constant.

of reference of the keyframe $\mathcal{K}_l$. The occupancy map is updated using both $\mathcal{Z}_l$ and the transformed $\mathcal{Z}_p$ through the standard logodds update [184]. If the voxel size α changes during the planning round, the occupancy grid map is regenerated using $\mathcal{Z}_l$ and $\mathcal{Z}_p$ using the new α. The time complexity of this regeneration step depends on the total number of voxels. In practice, we decide the number of voxels based on the available compute and keep them fixed throughout teleoperation.

The local occupancy map $\mathbf{m}$ partitions the space $\mathbb{R}^3$ into three subspaces: (1) free space $\mathcal{W}_{\text{free}}$, (2) occupied space $\mathcal{W}_{\text{occ}}$, and (3) unknown space $\mathcal{W}_{\text{unk}}$. To ensure safety, the motion plans sent to the robot must lie in free space, $\boldsymbol{\xi}_t \in \mathcal{W}_{\text{free}}$, for all time t. To check the motion plans against $\mathcal{W}_{\text{unsafe}} = \{\mathcal{W}_{\text{occ}} \cup \mathcal{W}_{\text{unk}}\}$, a common strategy is to compute a discrete distance field $\mathbf{d}$ where each point in the field stores the shortest distance to $\mathcal{W}_{\text{unsafe}}$. We use the variant of the fast-marching method by Sethian [164] over occupancy grids to compute the distance map $\mathbf{d}$ from the local map $\mathbf{m}$. It is assumed that the robot can fit a cube of side-length $2 \cdot r_{\text{robot}}$ and a r_{coll} amount of tolerance from $\mathcal{W}_{\text{unsafe}}$ is required for safety.

5.1.2 Motion Primitive Design

We use the control input parameterization by Yang et al. [203] that maps the space of joystick inputs to a finite set of forward-arc motion primitives. The joystick input is represented as $\mathbf{a}_t = \{v_{x,t}, v_{z,t}, \omega_t\}$, where $v_{x,t}$ is the velocity command in the $x_{\mathcal{B}}$ direction, $v_{z,t}$ is the velocity command in the $z_{\mathcal{B}}$ direction, and ω_t is the angular velocity command around the $z_{\mathcal{B}}$ direction. All velocity commands are uniformly dense sets clamped with user-specified bounds: $v_{x,t} \in [-V_x, V_x]$, $v_{z,t} \in [-V_z, V_z]$,

and $\omega_t \in [-\Omega, \Omega]$. Assuming a user-specified duration of the motion primitive, T, a forward-arc motion primitive $\gamma_t = \{\mathbf{a}_t, T\}$ can be generated by propagating the unicycle model [146]. The motion primitive γ_t is checked for feasibility and sent to the controller for execution.

Prior motion primitives-based teleoperation frameworks that utilize forward-arc motion primitives assume velocity command bounds to be constant and user-specified [203, 165]. These bounds can influence the design of the motion primitives when they are used for navigation in unknown environments. Such design decisions are made to ensure that the robot never enters an inevitable collision state [93]. For example, setting the $x_{\mathcal{B}}$-velocity bound, V_x, depends on many factors: (1) mapping time (Δt_{m}), (2) planning time (Δt_{p}), (3) sensing time (Δt_{s}), (3) sensing range ($z_{\max}$), (4) collision tolerance distance (r_{coll}), (5) robot radius (r_{robot}), and (6) maximum $x_{\mathcal{B}}$-deceleration (A_x) [73]. For the teleoperation task, we also need to account for the bounding box $\mathbf{B}$ of the local occupancy map $\mathbf{m}$. In terms of the voxel size and the number of voxels, the corners of the local map bounding box $\mathbf{B}$ are given by $\mathbf{b}_{\min} = -(\alpha/2) \cdot \{N_x, N_y, N_z\}$ and $\mathbf{b}_{\max} = (\alpha/2) \cdot \{N_x, N_y, N_z\}$. Since the robot is at the center of $\mathbf{B}$, the map information available in front of the robot is up to a distance $z_{\mathrm{eq}} = \min(z_{\max}, (\alpha/2)N_x)$ from the robot. Thus, an ideal upper bound for the maximum $x_{\mathcal{B}}$-velocity is derived using Euler motion equations as:

$$V_x = A_x \left(\sqrt{\Delta t_{\mathrm{l}}^2 + 2\frac{z_{\mathrm{eq}} - (r_{\mathrm{robot}} + r_{\mathrm{coll}})}{A_x}} - \Delta t_{\mathrm{l}} \right), \tag{5.1}$$

where, $\Delta t_{\mathrm{l}} = \Delta t_{\mathrm{s}} + \Delta t_{\mathrm{m}} + 2\Delta t_{\mathrm{p}}$. This equation represents an ideal upper bound because it assumes the deceleration A_x is attained instantly and does not consider the motor dynamics. Therefore, in practice, we reduce V_x with a constant δv to account for these unmodeled factors.

Note that Eq. (5.1) represents the velocity bound in terms of the voxel size α of the local occupancy map $\mathbf{m}$. Thus, the velocity bound should scale according to the resolution of the map. Consequently, the motion primitive design is dependent on the voxel size. We use this fact to adapt voxel size and the motion primitive design across and within planning rounds through hierarchical collision avoidance (Sect. 5.1.3).

5.1.3 Hierarchical Collision Avoidance

Algorithm 1 shows the pseudocode for the Hierarchical Collision Avoidance (HCA) algorithm. Instead of using a fixed voxel size α for the local occupancy map $\mathbf{m}$ throughout teleoperation, it is adapted hierarchically based on the output of the collision checker. At the start of the planning round, the voxel size α is set $\Delta\alpha$ above the one used in the previous planning round, α_{prev}, but clamped by a pre-specified $\alpha_{\max}$ and $\alpha_{\min}$ (Line 2). The motion primitive design is updated for this voxel size (Sect. 5.1.2, Line 5). The joystick input is mapped to the closest motion primitive, γ_{sel}, via grid search [203] (Line 6). A stopping motion primitive, γ_{stop}, is generated as a fallback action in case the next planning round fails (Line 7). The distance field (Sect. 5.1.1, Line 9) generated via the local map at the voxel size α (Sect. 5.1.1,

Algorithm 1: Hierarchical Collision Avoidance

```
1  function HCA(ξ_{t+Δt_p}, a, α_prev)
       input: ξ_{t+Δt_p}, a, α_prev
       parameters: α_max, α_min, Δα, Δt_p, Δt_m, Δt_s, N_x, N_y, N_z
       output: γ_sel, γ_stop, α_prev
2      α ← min(max(α_prev + Δα, α_min), α_max)
3      Δl ← 0
4      while Δl ≤ 2 do
5          V_x ← MaxSpeed (m, Δt_p, Δt_m, Δt_s)
6          γ_sel ← MapJoystick (ξ_{t+Δt_p}, a, V_x)
7          γ_stop ← StoppingAction (γ_sel, Δt_p)
8          m ← LocalMap (α, N_x, N_y, N_z)
9          d ← DistanceField (m)
10         if InCollision (γ_sel, γ_stop, d) then
11             α ← min(max(α − Δα, α_min), α_max)
12             Δl ← Δl + 1
13         else
14             break
15         end
16     end
17     α_prev ← α
18     if Δl > 2 then
19         return α_prev
20     else
21         return γ_sel, γ_stop, α_prev
22     end
23 end
```

Line 8) is used for collision checking. If either γ_{sel} or γ_{stop} are in collision (Line 10), we reduce the voxel size by $\Delta\alpha$ (Line 11), and try planning again. The map generation step contributes the most to the time complexity of this algorithm. If a feasible plan is found within a maximum of three map level changes (Line 18), it is returned (Line 21) and the next plan is executed. We choose to limit the checks to three map levels at a time to keep a consistent time complexity across planning rounds. If a feasible plan is still not possible, the current voxel size is returned as α_{prev} (Line 19) to use in the next planning round, and the previously planned stopping action is executed.

5.1.4 Implementation Detail

The proposed framework is deployed to the aerial system of dimensions 0.6 m × 0.3 m × 0.3 m and mass 2.5 kg (Fig. 5.3). This design is an improved version of the robot from [180] with a higher motor constant, longer flight time, and larger depth sensing range (Intel Realsense D455). The onboard companion computers, state estimation

Figure 5.3: The robot used in the field experiments is equipped with a forward-facing Intel Realsense D455, downward-facing mvBluefox global shutter color camera, and Pixracer flight controller.

Parameter	Value	Parameter	Value
Δt_{p}	0.1 s	N_y	20
Δt_{m}	0.08 s	N_z	20
Δt_{s}	0.07 s	$z_{\max}$	10.0 m
$\Delta\alpha$	0.01 m	r_{robot}	0.3 m
N_x	40	r_{coll}	0.1 m

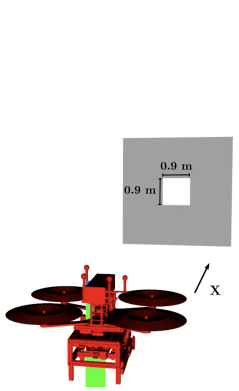
Table 5.1: Parameters common to all experiments.

system, the control system remain the same.

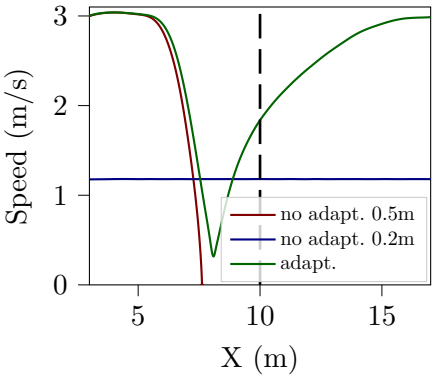
The parameters common to all the experiments are listed in Table 5.1. These parameters remain constant throughout teleoperation. Note that the operator does not need to specify a maximum velocity parameter found in most multirotor motion planning frameworks. Instead, the maximum velocity is determined in each planning round based on the local map extents (Sect. 5.1.2). Note that the $\Delta\alpha$ parameter specifies the change in map resolution after each collision checking iteration. Thus, this parameter should be specified based on the desired planning rate and the available onboard compute.

5.2 Results

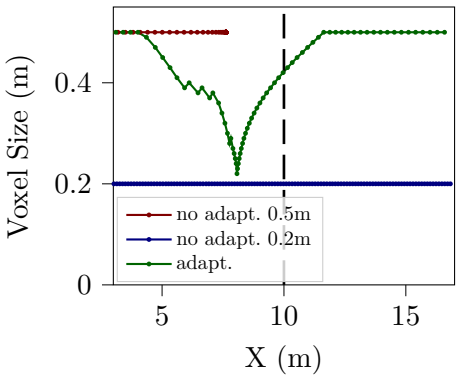
The adaptive teleoperation framework with hierarchical collision avoidance is implemented with a single thread on a CPU in C++ with the Robot Operating System (ROS) middleware. A 3.7 GHz Intel Core i9-10900K CPU with 32 GB RAM is used for the simulation experiments. A 1.8 GHz Intel Core i7-8550U CPU with 32 GB RAM is used for the hardware experiments. We evaluate the approach with four



(a) Window scenario



(c) Linear speed over distance



(d) Voxel size over distance

(b) Joystick

Figure 5.4: Performance comparison for the simulated window teleoperation task. (a) depicts the initial conditions for the task. A multirotor hovers at a distance of 10 m from a window of dimensions 0.9 m × 0.9 m. The operator controls the multirotor via the joystick shown in (b). The operator intends to go forward at the highest speed possible. (c) and (d) show the variation of the forward speed and the local map voxel size as a function of the distance from the window for the three teleoperation approaches.

teleoperation scenarios, two for simulation experiments (*Window Scenario*, *Varying-Clutter Cave Scenario*) and two for hardware experiments (*Door Scenario*, *Cave Scenario*). Experimentation for the Cave Scenario occurred at a cave on the Barbara Schomer Cave Preserve in Clarion County, PA.

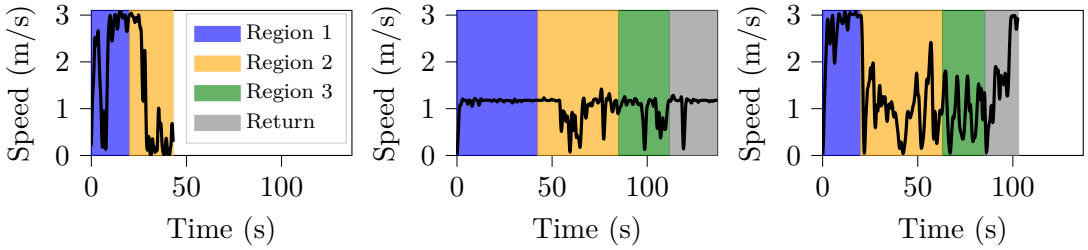
5.2.1 Simulation Experiments

Window Scenario: A multirotor is placed 10 m away from a simulated window of dimensions 0.9 m × 0.9 m (Fig. 5.4a). The operator intends to fly the multirotor in the forward direction, through the window, at the maximum possible speed (Fig. 5.4b). The teleoperation task is successful if the multirotor passes through the window without collisions and without the operator having to lower the raw joystick input. Three teleoperation methodologies are compared: (1) the proposed adaptive approach with $\alpha_{\max} = 0.5\,\text{m}$ and $\alpha_{\min} = 0.1\,\text{m}$; (2) a non-adaptive approach with a fixed voxel size, $\alpha_1 = 0.2\,\text{m}$; and (3) a non-adaptive approach with a fixed voxel size, $\alpha_2 = 0.5\,\text{m}$. These parameters are chosen such that for $\alpha_1 = 0.2\,\text{m}$ the local map is fine enough for the window to be visible while $\alpha_2 = 0.5\,\text{m}$ leads to a local map that is too coarse for it.

The results for the *Window Scenario* are shown in Fig. 5.4. We plot the speed and the local map voxel size over the X coordinate with respect to the window ($X = 0$)

(a) Simulated cave with varying clutter

(b) Speed heatmap for adaptive method



(c) No Adaptation 0.5 m

(d) No Adaptation 0.2 m

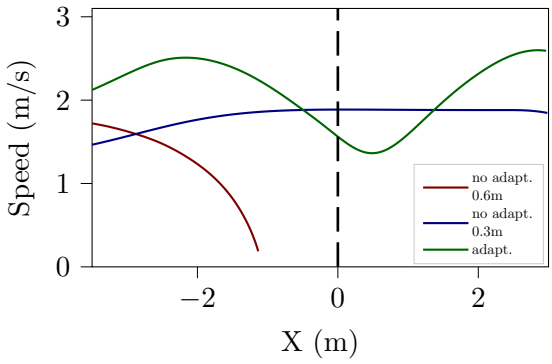
(e) Adaptation

Method	Region 1		Region 2		Region 3		Return	
	Time (s)	Av. Speed (m/s)	Time (s)	Av. Speed (m/s)	Time (s)	Av. Speed (m/s)	Time (s)	Av. Speed (m/s)
No adapt. 0.5 m	**20.0**	2.22	–	–	–	–	–	–
No adapt. 0.2 m	42.4	1.15	43.2	0.98	26.7	**0.97**	25.3	1.13
Adaptation	**20.0**	**2.67**	**43.0**	**1.02**	**22.3**	0.93	**17.7**	**1.69**

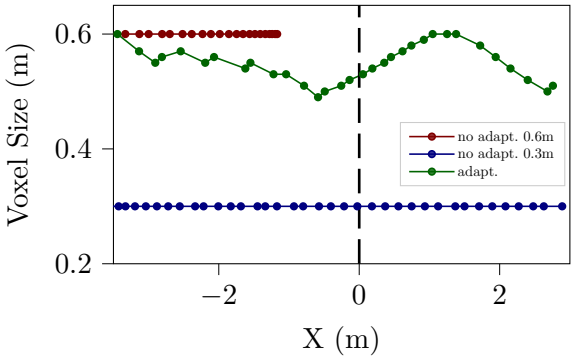
(f) Comparison of teleoperation statistics

Figure 5.5: Performance comparison for the (a) varying-clutter cave scenario with three different spaces: Region 1 is an open space, Region 2 is cluttered, and Region 3 is a narrow passage. The speeds achieved by the robot for each method are plotted over time in (c), (d), and (e). The graphs and the table in (f) demonstrate that the (c) No Adaptation 0.5 m variant cannot complete the circuit, while both the (d) No Adaptation 0.2 m and (e) Adaptation variants successfully traverse all regions. Our method completes the circuit in the least time while modulating speeds, as illustrated in the heatmap in (b). A video of this experiment can be found at https://youtu.be/VjyoPVXT8WY.

(a) Door scenario



(b) Linear speed over distance



(c) Voxel size over distance

Figure 5.6: Performance comparison for the door teleoperation task. (a) a robot starts at hover from outside a building and the operator intends to enter the building at the maximum possible forward speed (Fig. 5.4b) through a door of width 0.9 m. (b) and (c) show the speeds and voxel sizes as a function of distance from the door. A video of this experiment can be found at https://youtu.be/VjyoPVXT8WY.

in Figs. 5.4c and 5.4d respectively. For the non-adaptive case with $\alpha_1 = 0.2\,\text{m}$ voxel size, the operator teleoperates the multirotor through the window without collision at a constant speed of 1.17 m/s. For the non-adaptive case with $\alpha_2 = 0.5\,\text{m}$, the operator is not able to teleoperate the multirotor through the window while achieving a top speed of 3.03 m/s in open space. For the adaptive case, the operator can teleoperate through the window while being able to achieve a top speed of 3.03 m/s in open space, automatically slowing down to move through the window, and attaining the same top speed after passing through the window. The automatic slow down is expected due to the adaptation in the local map voxel size from $\alpha_{\text{max}} = 0.5\,\text{m}$ down to 0.25 m. Thus, the proposed approach allows speed modulation without requiring the operator to adjust the joystick input. Note that the speed attained close to the window in the adaptive case is lower than the constant speed of 1.17 m/s achieved in the non-adaptive 0.2 m voxel size case. This is because the length of the motion primitive when the robot is decelerating is longer than the case when the robot is moving at a constant speed. The collision checker marks the longer motion primitive infeasible, which results in a speed modulation that decelerates the robot to lower than 1.17 m/s.

This behavior may be changed through a collision checker that utilizes the higher derivatives of position, but this is left as future work.

Varying-Clutter Cave Scenario: A multirotor is placed in a simulated cave environment containing varying amounts of clutter (Fig. 5.5a). Region 1 is an open space, so the expected operation is high-speed multirotor teleoperation. Region 2 is a cluttered space where speeds must be reduced to ensure safety. Region 3 contains a narrow entrance to a larger passage that must be visible in the local occupancy map to enable robot access. The teleoperation task is to fly from Region 1 to Regions 2 and 3 and return to Region 1 (Fig. 5.5b). The operator supplies the directional inputs from the joystick but the forward speed input is always the maximum value. The three teleoperation methods from the *Window Scenario* are used here without modification in parameters. The methods are compared based on: (1) whether they allow access to all three regions and (2) the total time taken for the teleoperation task. Figures 5.5c–5.5e show the variation of the speed over time for each of the methods. Without adaptation, the method with a high voxel size can traverse Region 1 at high speeds and can enter Region 2 partially. However, it is unable to access the other regions. When the voxel size is lower or adaptive, the operator can teleoperate through all regions and return to Region 1. However, without adaptation, the speeds are lower compared to the adaptive-speed case. Consequently, the total time taken for the non-adaptive 0.2 m case is 137 s versus 103 s for the adaptive-speed case. Figure 5.5f shows the distribution of the time taken with the different phases of the teleoperation task. The adaptive-speed method achieves lower times and higher average speeds in most phases. These results demonstrate the efficacy of the proposed method in environments with varying amounts of clutter.

5.2.2 Real-World Experiments

Door Scenario: A multirotor is hovering outside a building, about 12 m away from a door, which has a frame with a 0.9 m width (Fig. 5.6a). The operator's intent and the measures of success are the same as in the *Window Scenario*, this time flying through the door to enter the building. The same teleoperation methodologies are compared as in the *Window Scenario*, with $\alpha_{\max} = 0.6\,\text{m}$, $\alpha_{\min} = 0.3\,\text{m}$, $\alpha_1 = 0.3\,\text{m}$, and $\alpha_2 = 0.6\,\text{m}$. The choice of these parameters is motivated by the same reasons as in the *Window Scenario*.

The results for the *Door Scenario* are shown in Fig. 5.6. Just like the *Window Scenario*, the speed and local map voxel sizes are plotted against the distance from the door ($X = 0$) in Figs. 5.6b and 5.6c, respectively. We observe similar results as in the *Window Scenario*. For the non-adaptive case with $\alpha_1 = 0.3\,\text{m}$, the operator can teleoperate through the door at a constant speed of 1.88 m/s. For the non-adaptive case with $\alpha_1 = 0.6\,\text{m}$, the operator is not able to teleoperate through the door while achieving high speeds outside the building. For the adaptive case, the operator can teleoperate through the door while being able to achieve a maximum speed of 2.50 m/s outside the building, automatically slowing down at the door and achieving the same maximum speed again after entering the building. Thus, the *Door Scenario* experiment demonstrates the efficacy of the proposed teleoperation method on the

Method	*Door Scenario*	*Cave Scenario*
No adapt. low vox. size	$0.06\,\mathrm{s} \pm 0.01\,\mathrm{s}$	–
No adapt. high vox. size	$0.03\,\mathrm{s} \pm 0.01\,\mathrm{s}$	$0.04\,\mathrm{s} \pm 0.01\,\mathrm{s}$
Adaptation	$0.06\,\mathrm{s} \pm 0.03\,\mathrm{s}$	$0.06\,\mathrm{s} \pm 0.03\,\mathrm{s}$

Table 5.2: Planning times for the real-world experiments.

computationally-constrained multirotor system shown in Fig. 5.3.

Cave Scenario: A multirotor hovers in a narrow passage inside of the wild cave shown in Fig. 1.1c. The operator intends to fly through the narrow passage without substantially altering the raw joystick input. Two teleoperation methodologies are compared in this case: (1) the adaptive approach with $\alpha_{\max} = 0.5\,\mathrm{m}$, $\alpha_{\min} = 0.1\,\mathrm{m}$ and (2) the non-adaptive approach with $\alpha = 0.5\,\mathrm{m}$.

The results for the *Cave Scenario* are shown in Fig. 5.7. In the no-adaptation case with $\alpha = 0.5\,\mathrm{m}$, the operator is not able to teleoperate the multirotor through the narrow passage of the cave due to the coarse local map. However, for the adaptation case, the voxel size is reduced automatically to allow for a finer local map for flight through the narrow passage. Thus, the proposed approach allows teleoperation through a narrow passage where it is difficult to guess the required voxel size of the local map before starting teleoperation.

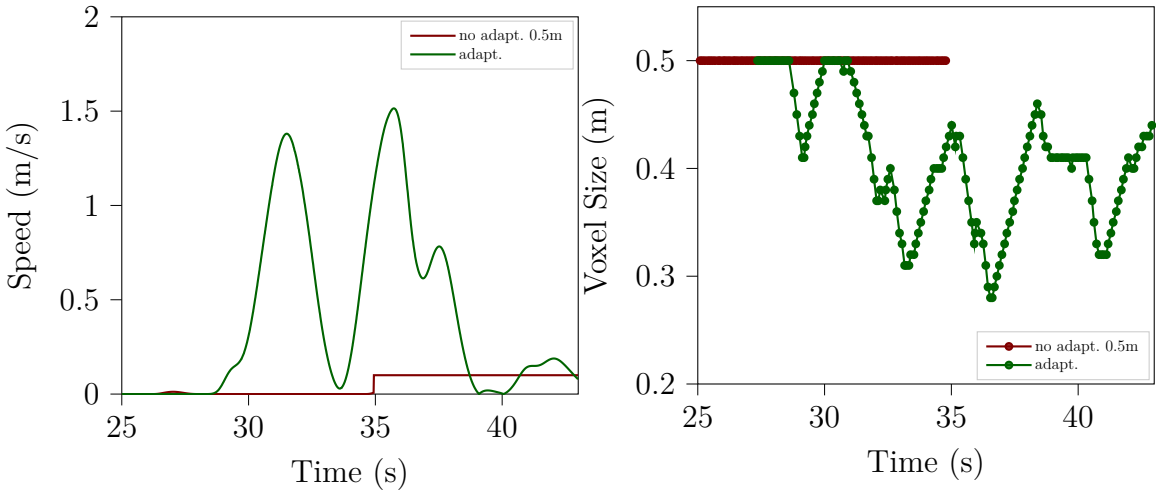
Table 5.2 illustrates the mean and standard deviation for total planning times for the real-world experiments. As noted in Sect. 5.1.3, most of the time complexity of the proposed approach is due to local map generation. Imposing an upper bound on the number of local map generation steps per planning round enables us to contain that time complexity and achieve a consistent planning rate (i.e., the low standard deviation in planning time).

5.3 Summary

In this chapter, we detail an approach for automatic maximum speed modulation (**R3**) for teleoperation of a multirotor in environments consisting of open, cluttered, and narrow spaces. We couple the motion primitive design and variable-resolution mapping to create a hierarchical collision avoidance method that modulates the maximum speed and voxel size of the local occupancy map simultaneously depending on the environment complexity. The framework is experimentally evaluated both in simulation and real-world complex environments, including caves, demonstrating that the speed and map resolution adaptation yields advantages both in terms of time taken and ability to complete a task.

Note that while this approach has been demonstrated for the teleoperation task, the same can be used for the reconstruction task in the preceding chapter with the local mapping strategy replaced with the one proposed in this chapter.

(a) Cave scenario



(b) Linear speed over time

(c) Voxel size over time

Figure 5.7: Performance comparison for the cave teleoperation task. (a) a robot starts at hover from a relatively spacious part of a cave passage. The operator intends to go through the passage at the maximum possible forward speed (Fig. 5.4b). (b) and (c) show the speeds and map voxel sizes as a function of time. Without map adaptation, the robot is unable to go through the narrow passage and the operator must land the robot around the 35 s mark. With map adaptation, the speeds are adapted according to the environment complexity and the robot traverses the narrow passage. A video of this experiment can be found at https://youtu.be/VjyoPVXT8WY.

CHAPTER 6

Adaptive Point Cloud Compression via Self-Organizing GMMs

This chapter presents the approaches addressing challenge **C3** which requires creating a mapping method that enables adaptive point cloud compression depending on the scene complexity for communication-efficient multi-robot reconstruction. Leveraging prior work in using GMMs for reconstruction to enable communication-efficiency for multi-robot operation [43], an adaptive point cloud compression strategy called *Self-Organizing Gaussian Mixture Modeling* (SOGMM) is presented that enables adaptive estimation of the GMM model complexity.

Section 6.1 details this approach. For convenience in the rest of this thesis, the model learnt through the SOGMM approach will be called a SOGMM model to distinguish from regular GMM models with fixed number of components. The experiments in Sect. 6.2 are designed for qualitative and quantitative comparison of the proposed approach and existing point cloud modeling techniques using publicly available simulated and real-world depth-intensity point clouds. A summary of this chapter is provided in Sect. 6.3.

6.1 Approach

Overview. An overview of the SOGMM system is shown in Fig. 6.1. We assume that a registered pair of depth, $\mathbf{I}_d \in \mathbb{R}^{h \times w}$, and grayscale, $\mathbf{I}_g \in \mathbb{R}^{h \times w}$, images $\mathcal{I} \coloneqq \{\mathbf{I}_d, \mathbf{I}_g\}$ of dimension $h \times w$ is provided for point cloud modeling. Note that the grayscale modality can also be thermal data scaled to a range $[0, 1]$. The image frame points may be projected to three dimension using the intrinsic camera matrix and associated with depth to obtain a pointcloud $\mathcal{Z}$ with hw points, where each point $\mathbf{x}_i \in \mathcal{Z}$ is four-dimensional and consists of 3D coordinates (x, y, z) augmented with a grayscale value g. Like prior works in GMM-based point cloud modeling [61, 166], we assume that the points in $\mathcal{X}$ are independently and identically distributed (i.i.d.) samples from an underlying continuous random variable X. The goal of the proposed approach is to model the joint probability distribution $p(x, y, z, g)$ as a GMM, $\mathcal{G}(\mathcal{I})$. Thus, the probability density for $\mathcal{G}(\mathcal{I})$ is written as:

$$\mathcal{G}(\mathcal{I}) \equiv p_X(\mathbf{x}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{6.1}$$

where, $\pi_m \in \mathbb{R}$, $\boldsymbol{\mu}_m \in \mathbb{R}^4$, and $\boldsymbol{\Sigma}_m \in \mathbb{R}^{4\times4}$ are the weight, mean, and covariance associated with the m^{th} multivariate Gaussian distribution component of the mixture. In contrast to existing techniques where the number of mixtures is specified *a priori* [179, 43] or utilize a hierarchy of GMMs to estimate the number of components [61, 166], the contribution of this work is a self-organizing approach that learns the number of mixtures from the underlying sensor data, $\mathcal{I}$, via information-theoretic techniques. This learned value is used for generative modeling for dataset $\mathcal{X}$ via the Expectation-Maximization (EM) algorithm [12], and utilizes steps (1a) through (1c) of K-Means++ [6] for initialization. The output of EM is the model $\mathcal{G}(\mathcal{I})$ from Eq. (3.1). Since this model is a joint distribution over the 3D spatial coordinates and the grayscale data, we can use the conditional distribution $p(g \mid x, y, z)$ to regress the grayscale image from the model for a given spatial point cloud and pinhole camera model [174, 166]. The remainder of this section describes how to estimate the number of mixture components M.

6.1.1 Principle of Relevant Information

The Principle of Relevant Information (PRI) [147] is an approach that extracts the relevant statistics from the dataset to learn a compressed representation of size M, equal to the number of locally-dense regions in the environment. The intuition behind PRI is to extract these relevant statistics by simultaneously minimizing the redundancy and distortion between the original and compressed datasets. Formally, let us consider a dataset, $\mathcal{Y}$, in which the points are assumed to be D-dimensional i.i.d. samples from a continuous multivariate random variable Y with the associated probability density function $p_Y(\mathbf{y})$. To create a compressed dataset $\mathcal{Y}_r$, with random variable Y_r and density $p_{Y_r}(\mathbf{y})$, the PRI is an information-theoretic optimization problem with the objective function:

$$J(Y_r) = \min_{Y_r} H_2(Y_r) + D_{\text{CS}}(Y_r, Y), \tag{6.2}$$

where, $H_2(Y_r)$ is the Renyi's quadratic entropy (RQE) of dataset $\mathcal{Y}_r$ calculated using the density $p_{Y_r}(\mathbf{y})$ and $D_{\text{CS}}(Y_r, Y)$ is the Cauchy-Schwarz divergence (CSD) between datasets $\mathcal{Y}_r$ and $\mathcal{Y}$ calculated using the densities $p_{Y_r}(\mathbf{y})$ and $p_Y(\mathbf{y})$. Minimizing RQE ensures less redundancy in the compressed dataset and minimizing CSD reduces the error induced due to compression. Principe [147] proves that when RQE and CSD have an equal contribution to the objective function in Eq. (6.2), the compressed dataset $\mathcal{Y}_r$ contains the modes of the original dataset $\mathcal{Y}$.

6.1.2 Gaussian Mean Shift

Rao et al. [152] show that the Gaussian Mean Shift (GMS) algorithm, as proposed by Cheng [35], is an iterative scheme for an approximate solution to the optimization problem in Eq. (6.2). The GMS algorithm uses a nonparametric estimate of the density

$p_Y(\mathbf{y})$,

$$p_Y(\mathbf{y}) = \frac{1}{|\mathcal{Y}|} \sum_{i=1}^{|\mathcal{Y}|} K_{\mathbf{H}}\left(\mathbf{y} - \mathbf{y}_i\right), \tag{6.3}$$

where, $\mathbf{H}$ is a symmetric positive definite $D \times D$ matrix and $K_{\mathbf{H}}\left(\mathbf{y} - \mathbf{y}_i\right)$ is a multivariate symmetric Gaussian kernel. This matrix is usually chosen proportional to the identity matrix $\mathbf{H} = \sigma^2 \mathbf{I}_{D \times D}$, where σ is called the *bandwidth* parameter [39]. Under this assumption the kernel $K_{\mathbf{H}}\left(\mathbf{y} - \mathbf{y}_i\right)$ simplifies to:

$$K_{\mathbf{H}}\left(\mathbf{y} - \mathbf{y}_i\right) = \frac{1}{(2\pi)^{D/2}\sigma} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{y}_i\|^2\right). \tag{6.4}$$

The compressed dataset $\mathcal{Y}_r$ containing the modes of the dataset is constructed through successive iterations using the update rule:

$$\mathbf{y}_{r,i}^t \leftarrow \frac{\sum_{j=1}^{|\mathcal{Y}|} K_{\mathbf{H}}\left(\mathbf{y}_{r,i}^{t-1} - \mathbf{y}_j^0\right) \mathbf{y}_j^0}{\sum_{j=1}^{|\mathcal{Y}|} K_{\mathbf{H}}\left(\mathbf{y}_i^{t-1} - \mathbf{y}_j^0\right)}, \tag{6.5}$$

where, the indices i and j iterate over each point in $\mathcal{Y}$ and the index $t \in \{0, \ldots, T-1\}$ is used to indicate a GMS iteration. At $t = 0$, the compressed dataset is initialized with the original dataset $\mathcal{Y}_r^0 = \mathcal{Y}^0$. The convergence criteria for GMS is based on the relative change in the points between successive iterations or the maximum number of iterations T are reached [28]. At the final iteration, the dataset contains many overlapping points (indicating modes) and is filtered based on Euclidean distances to obtain the final output $\mathcal{Y}_r$.

6.1.3 Mean Shift on Image Pair

For the SOGMM system, the value of M for a given image pair $\mathcal{I}$ as the number of unique points in the reduced dataset, i.e., $M = |\mathcal{Y}_r|$. In contrast to applying the mean shift algorithm in $\mathbb{R}^4$ using the point cloud $\mathcal{Z}$, we create the dataset $\mathcal{Y}$ from Eq. (6.2) in $\mathbb{R}^2$ space using depth and grayscale values from the image pair $\mathcal{I}$. Each point $\mathbf{y}$ in the dataset is a tuple (d_i, g_i), such that $d_i \in \mathbf{I}_d$, $g_i \in \mathbf{I}_g$, and i is an index for a pixel coordinate in the images. Thus, the size of the dataset $\mathcal{Y}$ is hw (equal to that of $\mathcal{Z}$) and $D = 2$.

The time complexity of the GMS algorithm increases quadratically with the number of points, linearly with the number of iterations, and linearly with the dimension of the data, $\mathcal{O}(TD|\mathcal{Y}|^2)$ [92]. The number of iterations for convergence T depends on the bandwidth value σ. In general, lower values of σ require larger values of T. For faster convergence, Carreira-Perpiñán [28] propose a modification to the update rule in Eq. (6.5) called the Gaussian Blurring Mean Shift (GBMS) algorithm and an early stopping criteria to obtain similar results. Instead of using the iterate $\mathbf{y}_j^0$ for every iteration in Eq. (6.5), GBMS uses the result from the previous iteration $\mathbf{y}_j^{t-1}$:

$$\mathbf{y}_{r,i}^t \leftarrow \frac{\sum_{j=1}^{|\mathcal{Y}|} K_{\mathbf{H}}\left(\mathbf{y}_{r,i}^{t-1} - \mathbf{y}_j^{t-1}\right) \mathbf{y}_j^{t-1}}{\sum_{j=1}^{|\mathcal{Y}|} K_{\mathbf{H}}\left(\mathbf{y}_i^{t-1} - \mathbf{y}_j^{t-1}\right)}. \tag{6.6}$$

Further, Comaniciu and Meer [39] show that utilizing a flat kernel as opposed to a Gaussian kernel also produces reasonable accuracy in applications like image segmentation while saving substantial computation due to the finite support of the flat kernel. As opposed to the Gaussian kernel in Eq. (6.4), the flat kernel is given by:

$$K_{\mathbf{H}}\left(\mathbf{y}-\mathbf{y}_i\right)=\begin{cases}1 & \text{if } \|\mathbf{y}-\mathbf{y}_i\|\leq\sigma\\ 0 & \text{if } \|\mathbf{y}-\mathbf{y}_i\|>\sigma\end{cases}. \tag{6.7}$$

In this work, we utilize both of these approximations to make PRI tractable for dense image pair data.

Figure 6.2 illustrates a result of applying the method on image pairs corresponding to a scene with low variation in depth and intensity (Fig. 6.2a) and a scene with high variation in depth and intensity (Fig. 6.2b). For increasing values of the bandwidth parameter σ used by the kernel in Eq. (6.5), we observe a monotonic decrease in the estimated number of components (Fig. 6.2c) and increase in the mean reconstruction error (Figs. 6.2d–6.2f). Further, for the scene in Fig. 6.2b, the estimated value of M is higher for all bandwidth values compared to the scene in Fig. 6.2a. This behavior is desired for the SOGMM system as the value of M in the model given by Eq. (3.1) must adapt automatically according to the scene complexity. Thus, using this system, adaptive complexity in GMM-based point cloud modeling can be achieved by only specifying the bandwidth parameter σ. The choice of bandwidth parameter can be based on the amount of computation available and the level of fidelity in the GMM model required by the application.

6.1.4 Incremental Multimodal Surface Mapping

Before describing the algorithm, three key data structures are described: the Local SOGMM, Global SOGMM, and Spatial Hash table.

Local SOGMM. A GMM model created via the SOGMM method using the relevant points corresponding to the latest multimodal point cloud. Each point $\mathbf{z}$ in this point cloud is assumed to be of the form, $\mathbf{z}=\{(\mathbf{x},i)\mid\mathbf{x}\in\mathbb{R}^3, i\in[0.0,1.0]\}$. Formally, the GMM model is represented as the function $\mathcal{G}_L\equiv\mathcal{G}_L(\mathbf{z})=\sum_{j\in\mathcal{J}}\pi_j\mathcal{N}(\mathbf{z}\mid\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)$, where π_j, $\boldsymbol{\mu}_j$, and $\boldsymbol{\Sigma}_j$ are the weight, mean, and covariance for the mixture component associated with index j in $\mathcal{G}_L$. Each mixture component is a Gaussian probability density $\mathcal{N}(\mathbf{z}\mid\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)$. The set of indices is denoted by $\mathcal{J}$. The sum of weights must be 1, $\sum_{j\in\mathcal{J}}\pi_j=1$, for a valid $\mathcal{G}_L$.

Global SOGMM. A GMM model created after merging all prior $\mathcal{G}_L$ models. The global model contains $|\mathcal{K}|\geq|\mathcal{J}|$ mixture components. Formally, $\mathcal{G}_G\equiv\mathcal{G}_G(\mathbf{z})=\sum_{k\in\mathcal{K}}\tau_k\mathcal{N}(\mathbf{z}\mid\boldsymbol{\nu}_k,\boldsymbol{\Lambda}_k)$, where different index and parameter symbols are used to notationally differentiate $\mathcal{G}_G$ from $\mathcal{G}_L$. Similar to $\mathcal{G}_L$, the set of indices is denoted by $\mathcal{K}$ and $\sum_{k\in\mathcal{K}}\pi_k=1$ must hold for a valid $\mathcal{G}_G$.

Spatial Hash. A hash table $H:\mathcal{M}\to\mathcal{Q}, m\mapsto\mathbf{q}^{\mathcal{A}}[h(m)]$ is created to map any point in 3D space (key $m\in\mathcal{M}$) to a vector of mixture component indices in $\mathcal{G}_G$ (value $\mathbf{q}^{\mathcal{A}}\in\mathcal{Q}$) that are within a fixed volume around the point. This fixed volume is

a cube with side length α. The hash function $h : \mathcal{M} \to \mathcal{A}$ maps the keys to an index set $\mathcal{A} = \{0, 1, \ldots, |\mathcal{A}| - 1\}$ used to insert into and query from H.

Figure 6.3 provides an overview of the multimodal surface mapping method. There are three steps: (1) creating $\mathcal{G}_L$, (2) merging $\mathcal{G}_L$ into $\mathcal{G}_G$, and (3) spatially hashing $\mathcal{G}_L$ into H. Details of each step are provided in the following sections.

6.1.4.1 Creating $\mathcal{G}_L$

For each point cloud $\mathcal{Z}$, the relevant subset of the point cloud $\mathcal{Z}^r$ is determined. If the number of points in this set is greater than a pre-specified threshold, then $\mathcal{G}_L$ is created; otherwise, $\mathcal{Z}^r$ is cached and used along with the subsequent frames. $\mathcal{Z}^r$ represents points not already modeled by $\mathcal{G}_G$. Therefore, if $\mathcal{G}_G$ is not initialized, all points are treated as relevant ($\mathcal{Z}^r = \mathcal{Z}$). Otherwise, $\mathcal{Z}^r$ is determined using a threshold (ϕ) on the log-likelihood [166] that points $\mathcal{Z}$ originated from the model $\mathcal{G}_G$,

$$\mathcal{Z}^r = \{\mathbf{z} \in \mathcal{Z} \mid \mathcal{L}(\mathbf{z}) = \ln\left(\mathcal{G}_G(\mathbf{z})\right), \mathcal{L}(\mathbf{z}) < \phi\}. \tag{6.8}$$

However, this approach has two drawbacks when used with multimodal point clouds. First, thresholding the log-likelihood scores via Eq. (6.8) over the multimodal point cloud directly does not yield the intended $\mathcal{Z}^r$ as the 4^{th} dimension contains intensity data, which is not in the metric space of the other three dimensions. Second, as the size of the model $\mathcal{G}_G$ (i.e., the value of K) increases over time, the time complexity of calculating Eq. (6.8) increases linearly. Performing this computation for all points in $\mathcal{Z}$ is prohibitive for real-time operation on computationally-constrained robotic systems.

The first challenge is addressed by utilizing the marginal probability density $p(\mathbf{x})$ instead of $p(\mathbf{z})$ for the log-likelihood calculation, i.e.,

$$\mathcal{Z}^r = \{\mathbf{z} \in \mathcal{Z} \mid \mathbf{z} = (\mathbf{x}, i), \mathcal{L}(\mathbf{x}) < \phi\} \text{ where,} \tag{6.9}$$

$$\mathcal{L}(\mathbf{x}) = \ln\left(\sum_{k \in \mathcal{K}} \tau_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\nu}_k^{\mathbf{x}}, \boldsymbol{\Lambda}_k^{\mathbf{xx}})\right), \tag{6.10}$$

$$\boldsymbol{\nu}_k = [\boldsymbol{\nu}_k^{\mathbf{x}}, \boldsymbol{\nu}_k^{i}]^\top, \text{ and } \boldsymbol{\Lambda}_k = \begin{bmatrix} \boldsymbol{\Lambda}_k^{\mathbf{xx}} & \boldsymbol{\Lambda}_k^{\mathbf{x}i} \\ \boldsymbol{\Lambda}_k^{i\mathbf{x}} & \boldsymbol{\Lambda}_k^{ii} \end{bmatrix}. \tag{6.11}$$

Figure 6.4 details the effect of using Eq. (6.9) instead of Eq. (6.8). The value of ϕ is determined empirically, as done in [167], and it is fixed for the synthetic and real-world scenarios.

The second challenge is addressed by selecting only the subset of mixture components from $\mathcal{G}_G$ (i.e., selecting $\mathcal{B} \subseteq \mathcal{K}$) that are overlapped by or close to the points in $\mathcal{Z}$. This way we can reduce the number of summands in Eq. (6.10). The hash table H is leveraged for this purpose. Each 3D point $\mathbf{x}$ from $\mathbf{z} \in \mathcal{Z}$ is a key m for the hash function h that is used to search H for the closest vector of mixture component indices, $\mathbf{q}^{\mathcal{A}}[h(m)]$. After attaining these vectors for all points in $\mathcal{Z}$, the unique set of mixture component indices form the index set $\mathcal{B}$. For the example scenario in Fig. 6.4, Fig. 6.4d shows the output after this approximation. The output is similar to the case when

the original set of components $\mathcal{K}$ (Fig. 6.4c) is used but $\mathcal{B}$ is smaller (480 elements instead of 1165). Consequently, for this example the time taken to compute Eq. (6.10) with $\mathcal{K}$ is 0.35 s whereas with $\mathcal{B}$ it is 0.25 s (28% faster). Note that the ratio $|\mathcal{K}|/|\mathcal{B}|$ grows over time as the size of $\mathcal{G}_G$ increases when point clouds from new regions are observed. An analysis of the computational savings using this approach is provided in Sect. 6.2.3.

6.1.4.2 Merging $\mathcal{G}_L$ into $\mathcal{G}_G$

After the $\mathcal{G}_L$ model is created using $\mathcal{Z}^r$, it is merged with the global point cloud model $\mathcal{G}_G$ by appending the parameters and re-normalizing the weights. Let the global model before merging be $\mathcal{G}_G^t$ and after merging be $\mathcal{G}_G^{t+1}$. The parameters for $\mathcal{G}_G^{t+1}$ are given by $\boldsymbol{\tau}^{t+1} = [\boldsymbol{\tau}^t, \boldsymbol{\pi}]^\top$ such that $\sum_{b\in\mathcal{B}} \tau_b^{t+1} = 1$, $\boldsymbol{\nu}^{t+1} = [\boldsymbol{\nu}^t, \boldsymbol{\mu}]^\top$, and $\boldsymbol{\Lambda}^{t+1} = [\boldsymbol{\Lambda}^t, \boldsymbol{\Sigma}]^\top$. The index set for the global model is also updated and the number of components increase accordingly, $|\mathcal{K}|^{t+1} = |\mathcal{K}|^t + |\mathcal{J}|$.

6.1.4.3 Spatially Hashing $\mathcal{G}_L$ into H

In addition to the global model, the hash table H is updated using the mixture components from the latest local model $\mathcal{G}_L$. A total of $|\mathcal{J}|$ hash keys are inserted into the table where each hash key m_j is the spatial part of the mean position $\boldsymbol{\mu}_j^{\mathbf{x}}$ along with points generated at constant probability ellipsoids corresponding to 68% (1-sigma), 95% (2-sigma) and 99.7% (3-sigma) of the data points, for all $j \in \mathcal{J}$.

The hash function h that maps m_j to an index in $\mathcal{A}$ is given by $h(m_j) \equiv h(\boldsymbol{\mu}_j^{\mathbf{x}}) = N_z(r(\boldsymbol{\mu}_j^{\mathbf{x}})N_x + c(\boldsymbol{\mu}_j^{\mathbf{x}})) + s(\boldsymbol{\mu}_j^{\mathbf{x}})$ such that, $r(\boldsymbol{\mu}_j^{\mathbf{x}}) = \lfloor(\boldsymbol{\mu}_j^y - \mathbf{o}^y)/\alpha\rfloor$, $c(\boldsymbol{\mu}_j^{\mathbf{x}}) = \lfloor(\boldsymbol{\mu}_j^x - \mathbf{o}^x)/\alpha\rfloor$, and $s(\boldsymbol{\mu}_j^{\mathbf{x}}) = \lfloor(\boldsymbol{\mu}_j^z - \mathbf{o}^z)/\alpha\rfloor$. Here, $[N_x, N_y, N_z]$ are the number of cells along each axis of a 3D regular grid of spatial resolution α, $\mathbf{o} = -\frac{1}{2}\alpha[N_x, N_y, N_z]$ is the origin position of this grid, and $\lfloor.\rfloor$ is the floor operator. Intuitively, the hash function h assigns the mean positions $\boldsymbol{\mu}^{\mathbf{x}}$ into a sparse grid of a pre-specified extent $[N_x, N_y, N_z]$ and resolution α.

The value corresponding to each key is the index of the component j in the global model after the merging step (Sect. 6.1.4.2) is complete. Thus, the value corresponding to the hash key m_j is $|\mathcal{K}|^t + j$. It is possible that multiple hash keys are mapped to the same cell in the grid (i.e., hash collisions are possible). An example scenario is when α is large and a subset of means $\boldsymbol{\mu}^{\mathbf{x}}$ are spatially within α distance. In this case, we want to store all the values in a vector. This is why the value set $\mathcal{Q}$ is defined as a set of vectors as opposed to a set of integers. If a hash collision occurs for any two keys m_f and m_g (i.e., $h(m_f) = h(m_g)$), the values are appended into the vector $\mathbf{q}^{\mathcal{A}}[h(m_f)]$.

6.1.5 Global Spatial and Intensity Inference

Given the global model $\mathcal{G}_G$, we want to reconstruct the environment spatially along with the intensity values. The marginal global model given by $\boldsymbol{\tau}$, $\boldsymbol{\nu}^{\mathbf{x}}$, and $\boldsymbol{\Lambda}^{\mathbf{xx}}$ (as

defined by Eq. (6.11)) is used for spatial inference and densely sampled using the Box-Muller transform [19]. The conditional probability density $p(i \mid \mathbf{x})$ (as noted in [76]), is used to infer intensity at the sampled spatial points. This inference is performed in batches of components from $\mathcal{G}_G$. The batch size is determined based on the available memory on the CPU used to perform inference.

6.2 Results

In this section, the performance of the SOGMM method is evaluated on real-world point cloud data provided by Zhou and Koltun [210]: `stonewall` (Fig. 6.5a), `copyroom` (Fig. 6.5b), and `lounge` (Fig. 6.5c). Because the proposed methodology addresses efficient and compact perceptual modeling within the robotics context, it is compared against three open source mapping baselines: (1) OctoMap (OM) [88], (2) NDTMap (NDT) [118], and (3) GMM with fixed number of components (FC) [180]. The voxel resolutions for the first two approaches are set to 0.02 m and 0.05 m, which have been demonstrated to be adequate for scene representation [202]. For this section only, the following shorthand is introduced: an OM with 0.05 m and 0.02 m leaf sizes will be referred to as OM-0.05 and OM-0.02, respectively; an NDTMap with voxel resolutions of 0.05 m and 0.02 m will be referred to as NDT-0.05 and NDT-0.02, respectively; the GMM approach with 75, 500, and 2000 components will be referred to as FC-75, FC-500, and FC-2000, respectively; and the SOGMM approach with bandwidths 0.01, 0.02, and 0.03 will be referred to as SOGMM-0.01, SOGMM-0.02, and SOGMM-0.03, respectively.

Reconstruction from Environment Models. For OctoMap, the reconstruction at the minimum leaf size is utilized after modeling occupied space using the color class[1]. For NDTMap, the reconstruction is obtained by modifying the `NDTCell` class to store the average grayscale value for the points associated with the cell[2]. In both cases, intensity is queried at a 3D coordinate. For the FC and SOGMM approaches, the 3D reconstruction is obtained by densely sampling the joint distribution $p_X(\mathbf{x})$. The grayscale reconstruction is obtained by regressing the expected value from the conditional distribution $p_{g|\mathbf{x}}(g|\mathbf{x})$. This conditional distribution is obtained as follows. First, for each component m in the model given by Eq. (3.1), the mean and covariance can be written as:

$$\boldsymbol{\mu}_m = \begin{bmatrix} \boldsymbol{\mu}_{m,\mathbf{x}} \\ \boldsymbol{\mu}_{m,g} \end{bmatrix} \quad \boldsymbol{\Sigma}_m = \begin{bmatrix} \boldsymbol{\Sigma}_{m,\mathbf{xx}} & \boldsymbol{\Sigma}_{m,\mathbf{x}g} \\ \boldsymbol{\Sigma}_{m,g\mathbf{x}} & \boldsymbol{\Sigma}_{m,gg} \end{bmatrix}.$$

The conditional distribution $p_{g|\mathbf{x}}(g|\mathbf{x})$ can be expressed using these quantities

[1] https://github.com/OctoMap/octomap/blob/devel/octomap/include/octomap/ColorOcTree.h

[2] https://github.com/OrebroUniversity/perception_oru/blob/port-kinetic/ndt_map/include/ndt_map/ndt_cell.h

as [167, 174]:

$$p_{g|\mathbf{x}}(g|\mathbf{x}) = \sum_{m=1}^{M} w_m(\mathbf{x}) \mathcal{N}(g \mid \lambda_m(\mathbf{x}), \nu_m^2), \tag{6.12}$$

where

$$w_m(\mathbf{x}) = \frac{\pi_m \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{m,\mathbf{x}}, \boldsymbol{\Sigma}_{m,\mathbf{xx}})}{\sum_{m'=1}^{M} \pi_{m'} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{m',\mathbf{x}}, \boldsymbol{\Sigma}_{m',\mathbf{xx}})},$$
$$\lambda_m(\mathbf{x}) = \boldsymbol{\mu}_{m,g} + \boldsymbol{\Sigma}_{m,g\mathbf{x}} \boldsymbol{\Sigma}_{m,\mathbf{xx}}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{m,\mathbf{x}}),$$

and

$$\nu_m^2 = \boldsymbol{\Sigma}_{m,gg} - \boldsymbol{\Sigma}_{m,g\mathbf{x}} \boldsymbol{\Sigma}_{m,\mathbf{xx}}^{-1} \boldsymbol{\Sigma}_{m,\mathbf{x}g}.$$

Finally, the expected value for Eq. (6.12) is:

$$\lambda(\mathbf{x}) = \sum_{m=1}^{M} w_m(\mathbf{x}) \lambda_m(\mathbf{x}). \tag{6.13}$$

6.2.1 Qualitative Evaluation

Figure 6.5 provides a qualitative evaluation of the OctoMap, NDTMap, and SOGMM methods by comparing the reconstruction of three scenes of varying complexity. Figures 6.5a–6.5c provide images of low, medium, and high complexity scenes, respectively. The OctoMap method does not allow smooth color representation at the boundaries of voxels, as shown in Figs. 6.5d–6.5f. The NDTMap representation can be sampled using the Gaussian components in each voxel. This results in a qualitatively better reconstruction (Figs. 6.5g–6.5i) compared to OctoMap at a cost of high memory footprint. The SOGMM representation is sampled to generate reconstructions shown in Figs. 6.5j–6.5l, respectively. While the bandwidth parameter remains constant across scenes, the estimated number of components increases with the scene complexity. The reconstructed *wall* retains the fine details in the intensity and edges around the individual stones. Note that in the reconstruction of the *copier* (Fig. 6.5k), the thin wires on the right-hand side of the image are accurately modeled and the intensity is retained. In Fig. 6.5l one can see the individual leaves and shadows of the *plant* are preserved between the original image and reconstruction. The SOGMM framework preserves fine details by learning the appropriate model complexity from the underlying sensor data without parameter tuning from the user.

6.2.2 Quantitative Evaluation

For quantitative evaluation of the reconstructed point clouds and grayscale images, we utilize three metrics: (1) Peak Signal-To-Noise Ratio (PSNR), (2) Mean Reconstruction Error (MRE), and (3) Memory Usage. The PSNR quantifies how accurately

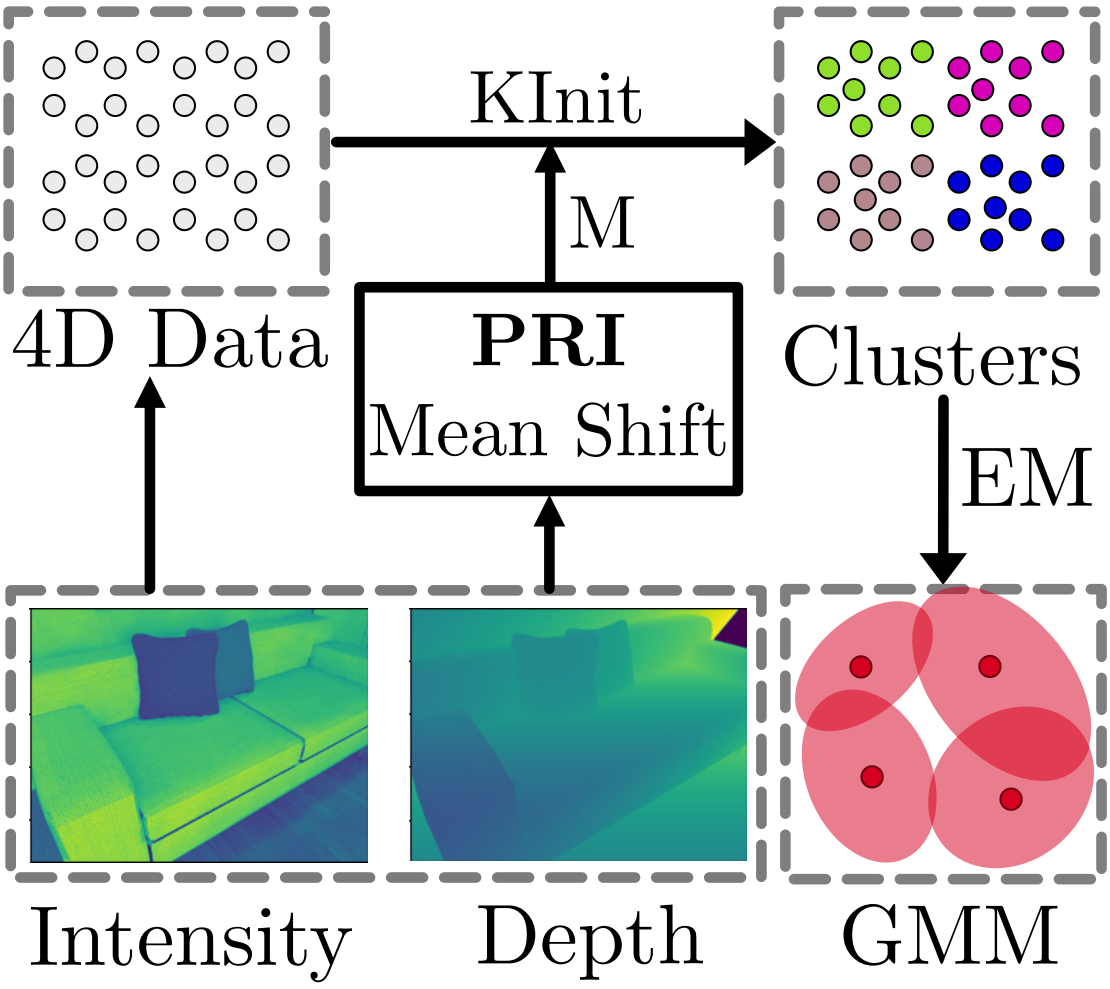
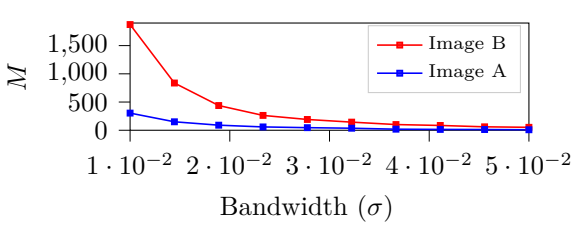


Figure 6.1: Overview of the proposed point cloud modeling method. The registered intensity-depth image pair is used by the PRI component to determine the number of modes M. Steps (1a)–(1c) of the K-Means++ algorithm [6] perform a hard-partitioning of the 4D data into M clusters (KInit). Finally, the EM algorithm does a soft-partitioning of the 4D data using the KInit output to create a M-component finite GMM. The proposed system encodes the 4D point cloud data into a finite GMM without requiring the specification of number of mixtures M for every image pair.

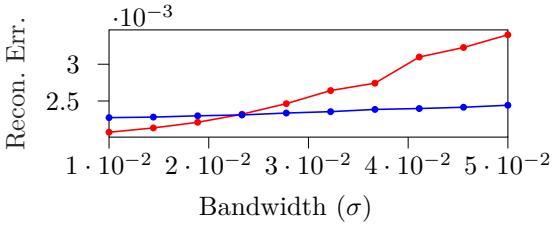
Dataset	OM-0.02			NDT-0.02				FC-2000				SOGMM-0.01			
	Recon Err.		Mem.	Recon Err.		Mem.		Recon Err.		Mem.		Recon Err.		Mem.	
	PSNR	(m)	(MB)	PSNR	(m)	(MB)	# Voxels	PSNR	(m)	(MB)	# Comp.	PSNR	(m)	(MB)	# Comp.
Wall	33.9	0.010	0.078	38.0	**0.002**	0.26	6482	**40.5**	**0.002**	0.12	2000	39.1	**0.002**	**0.06**	933
Copier	25.8	0.010	**0.10**	30.0	**0.002**	0.31	7856	**33.4**	**0.002**	0.12	2000	33.0	**0.002**	**0.10**	1599
Plant	26.3	0.010	**0.10**	30.8	**0.002**	0.30	7470	35.5	0.002	0.12	2000	**36.0**	**0.002**	0.15	2464

Table 6.1: Raw data for the quantitative comparison of the OM-0.02, NDT-0.02, FC-2000, and SOGMM-0.01 cases. The SOGMM method achieves a balance between the model fidelity and size for diverse scenes without changing the single tunable parameter σ.

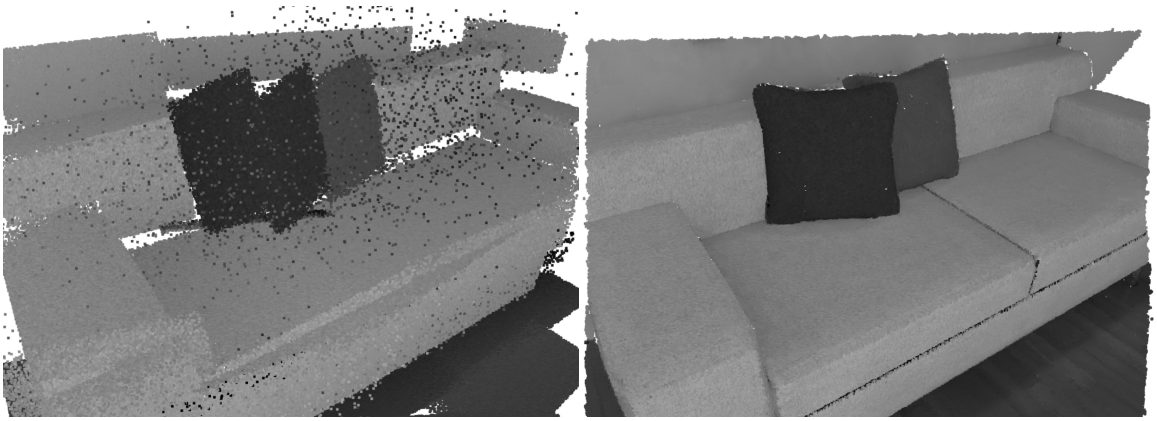
(a) Image A (b) Image B



(c) Estimated Number of Components



(d) Recon. Err. Variation



(e) Reconstruction, $\sigma = 0.05$ (f) Reconstruction, $\sigma = 0.01$

Figure 6.2: A study on how the PRI (Eq. (3.1)) component in the SOGMM system (Fig. 6.1) adapts the model size according to the scene complexity. A simple scene consisting of (a) homogeneous, white walls requires fewer components than a (b) complex scene consisting of discrete, structured objects. (c) plots the number of components required to represent each of the two scenes for a given bandwidth parameter, σ. (d) plots the mean reconstruction error variation with σ. (e) and (f) show the reconstruction result for the extrema bandwidths in (d). Note how the SOGMM formulation selects more components to represent the complex scene for a given bandwidth value. Further, the reconstruction error varies monotonically with σ.

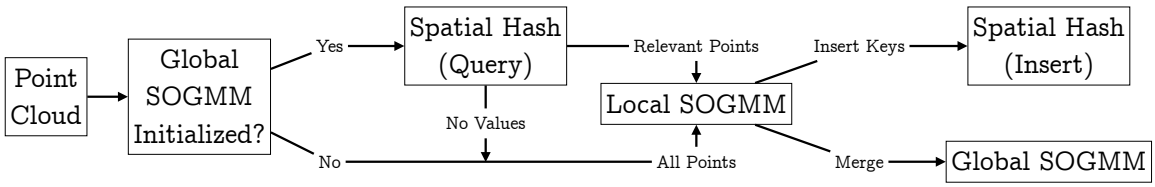


Figure 6.3: Information flow during surface point cloud modeling via the proposed incremental mapping approach (Sect. 6.1.4).

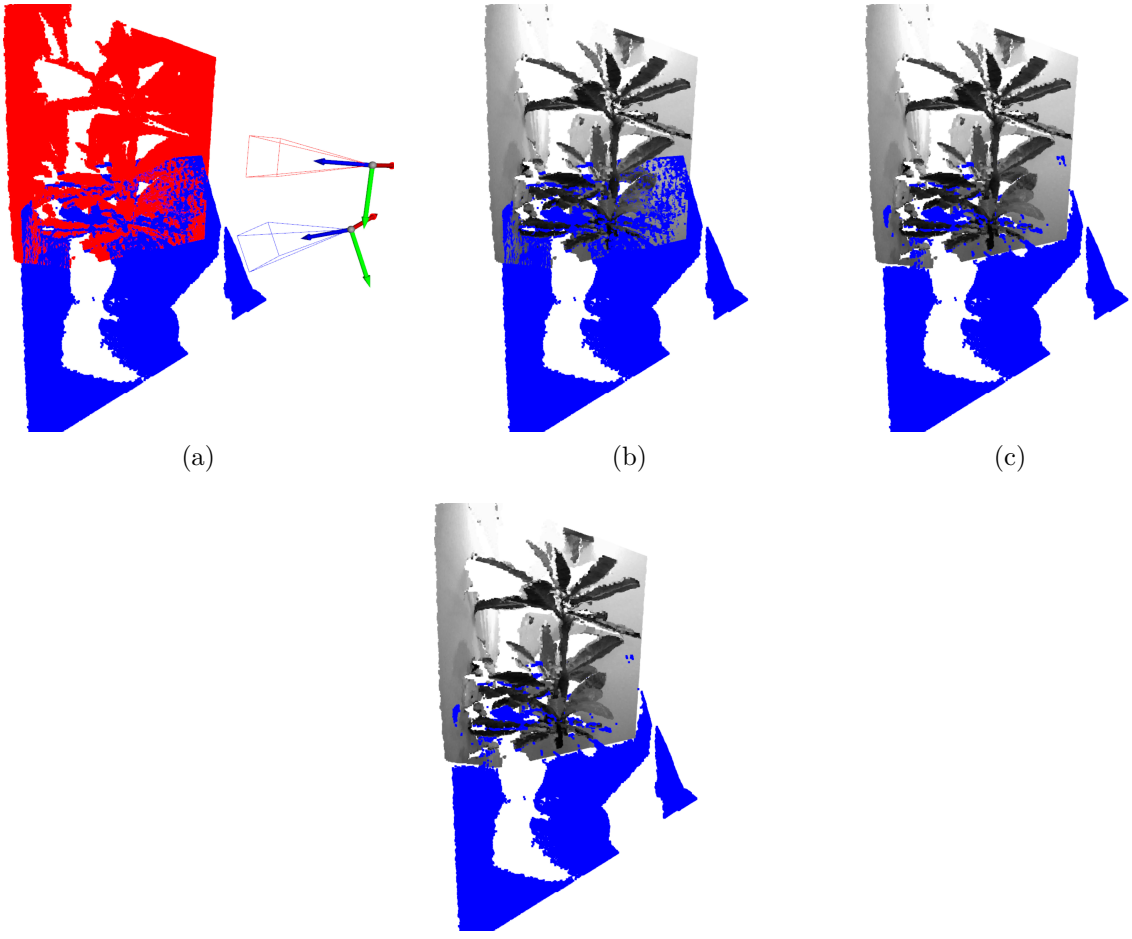


Figure 6.4: Illustration of the relevant point cloud calculation using two multimodal point clouds, $\mathcal{Z}_1$ and $\mathcal{Z}_2$ (Sect. 6.1.4.1). The objective is to find the relevant point cloud, $\mathcal{Z}_2^r$, from $\mathcal{Z}_2$ using $\mathcal{G}_G$, which is created from $\mathcal{Z}_1$. (a) shows the 3D parts of these point clouds in different colors and the associated 3D poses. (b) shows $\mathcal{Z}_2^r$ and $\mathcal{Z}_1$ with intensity values, when Eq. (6.8) is used. (c) shows the same but when Eq. (6.9) is used. Notice that in the former case $\mathcal{Z}_2^r$ contains more misclassified points that overlap with $\mathcal{G}_G$ than in the latter case. (d) shows the output $\mathcal{Z}_2^r$ when only a subset ($|\mathcal{B}| = 480$) of components in $\mathcal{G}_G$ ($|\mathcal{K}| = 1165$) derived using the hash table H are used. This output is similar to (c). The point clouds are sourced from from the real-world Lounge dataset [210]. *This figure is best viewed in color.*

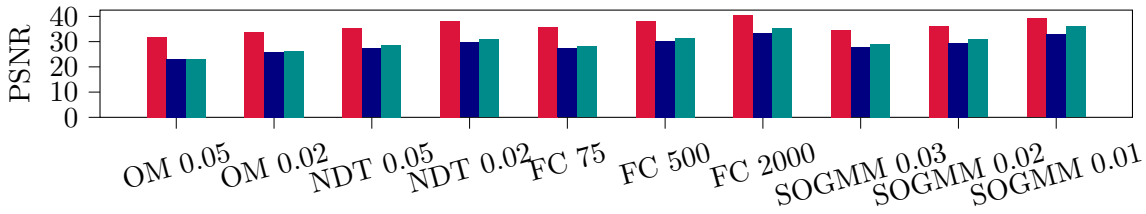
(a) *Wall* (original) (b) *Copier* (original) (c) *Plant* (original)

(d) OctoMap 0.02 m (e) OctoMap 0.02 m (f) OctoMap 0.02 m

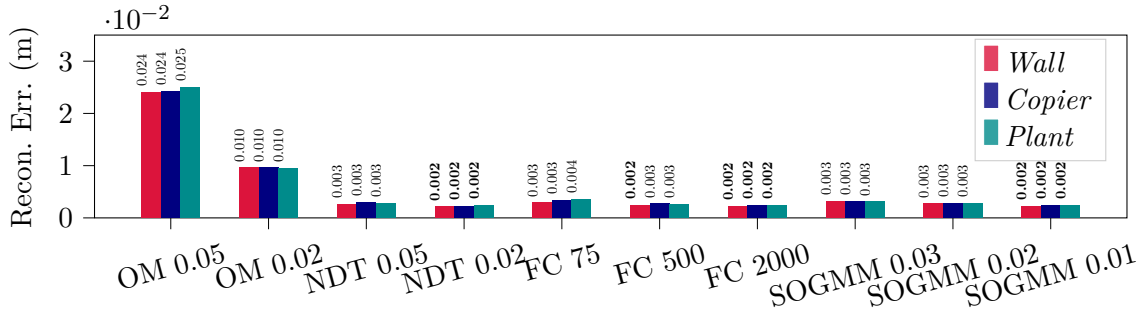
(g) NDTMap 0.02 m (h) NDTMap 0.02 m (i) NDTMap 0.02 m

(j) $\sigma = 0.01$ (k) $\sigma = 0.01$ (l) $\sigma = 0.01$

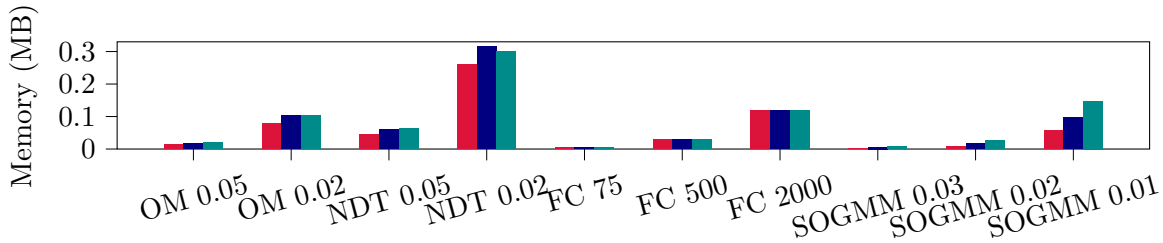
Figure 6.5: Resampled output from SOGMMs created for three point clouds with different levels of complexity. The point clouds are taken from real-world datasets [210]. The OctoMap method results in a pixelated output. NDTMap allows a smoother output at a cost of high memory usage. The SOGMM method adapts the complexity of the mixture model without changing parameters across different scenes ($\sigma = 0.01$ for all the cases). A supplementary video may be found at https://youtu.be/TCn5KB1m5P0.



(a) PSNR for Grayscale Image Reconstruction (higher is better)



(b) Mean Reconstruction Error for 3D Reconstruction (lower is better)



(c) Size of the Model (lower is better)

Figure 6.6: Quantitative evaluation of the SOGMM method for the scenes shown in Fig. 6.5. The SOGMM approach enables high accuracy (a) grayscale and (b) 3D reconstructions while allowing (c) adaptation in the size of the model without changing the bandwidth parameter across scenes.

intensity is reconstructed from the model by comparing against the original intensity image. The MRE is computed for the spatial part of the reconstructed point cloud (i.e., $[x, y, z]^\top$) and is given by the average distance between the reconstructed point cloud and the ground truth point cloud. The memory usage quantifies the memory required to store the model in megabytes (MB). For OctoMap, this value is the size of the `.ot` file that retains the occupancy and grayscale information. Each cell of the NDTMap stores a Gaussian component and intensity, so the memory usage (assuming floating point values and M cells) is calculated as $4 \cdot M \cdot (1 + 3 + 6)$ bytes. The FC and SOGMM are composed of M 4D Gaussian components, so the memory usage is calculated as $4 \cdot M \cdot (1 + 10 + 4)$ bytes.

Figure 6.6 shows these metrics for the baselines and our approach for three scenes (*Wall*, *Copier*, and *Plant*) shown in Fig. 6.5. The OM baseline exhibits lower PSNR values (Fig. 6.6a) and highest reconstruction error (Fig. 6.6b) compared to other approaches. NDT-0.05 and NDT-0.02 outperform OM-0.05 and OM-0.02 in terms of

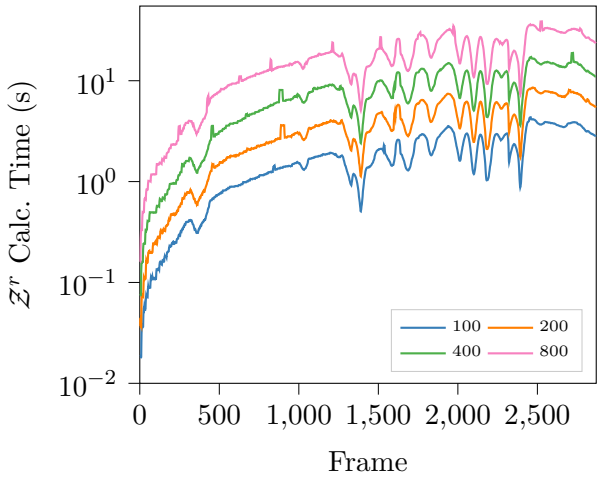
PSNR and reconstruction error but consume larger amounts of memory (Fig. 6.6c).

While FC-75, FC-500, and FC-2000 consume less memory compared to NDT-0.02, they are each constrained to leverage the same representational fidelity across the three scenes without consideration for the scene complexity. Without the ability to adapt, homogenous scenes will be represented with more fidelity than required and complex scenes will have insufficient fidelity, which is undesirable. For example, a large number of components may not be required to achieve good PSNR and reconstruction error scores for the *Wall* scene due to its low complexity; however, FC-2000 is required to use 2000 components *a priori*. In contrast, SOGMM cases automatically decide to use lower number of components for *Wall* and progressively higher number of components for *Copier* and *Plant*. Consequently, we observe similar PSNR and reconstruction error scores for SOGMM-0.01 and FC-2000; however, the former case adapts to the scene and consumes less memory for the *Wall* and *Copier* cases. Table 6.1 provides raw evaluation data to directly compare SOGMM-0.01 against the best performing variants of OctoMap, NDT, and FC. SOGMM-0.01 produces a model with accuracy in line with NDT-0.02, while consuming significantly less memory. FC-2000 allows higher PSNR scores but overestimates the number of components for scenes with relatively less intensity-depth variation. SOGMM-0.01 on the other hand provides a variation in the number of components used without changing any parameters across the three scenes.

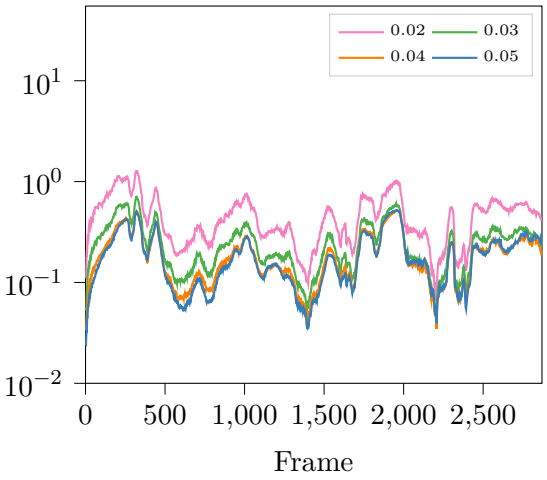
SOGMM-0.02 and SOGMM-0.03 show the effect of the bandwidth parameter σ. As expected from Fig. 6.2, the size of the model increases σ decreases. The σ parameter can be chosen based on the available computation as opposed to making an *a priori* guess about the complexity of the environments being modeled. The experimental results are divided into two parts. In Sect. 6.2.3, the computational performance gain offered by the proposed incremental mapping approach due to the spatial hash formulation is compared with the prior work on GMM-based multi-modal mapping [166]. In Sect. 6.2.4 the reconstruction accuracy from the global map created through the proposed approach is compared with Octomap [88], Voxblox (Nvblox[3]) [137], and GMM-based maps that use a fixed number of components (FCGMM) [180]. These methods are chosen as baselines because they have been used for our target application, multi-robot 3D reconstruction. We will use the "Method-Parameter" notation to denote the parameter being used. For example, Proposed-0.02 denotes the proposed approach with the bandwidth parameter $\sigma = 0.02$.

One synthetic (D1: Living Room from the Augmented ICL-NUIM datasets [36]) and three real-world datasets (D2: Lounge, D3: Copyroom, and D4: Stonewall from the Redwood datasets [210]) are used for qualitative and quantitative evaluation of the proposed approach and its comparison with the baseline methods. All datasets contain 640×480 RGB and depth images along with the corresponding camera poses. All methods are provided reduced resolution 128×96 images for incremental mapping. The computer used for all experiments contains an Intel Core i9-10900K CPU (20 threads, maximum clock speed 5.3 GHz, 32 GB RAM) and a NVIDIA GeForce RTX

[3]Nvblox is the GPU-accelerated extension of Voxblox: https://github.com/nvidia-isaac/nvblox



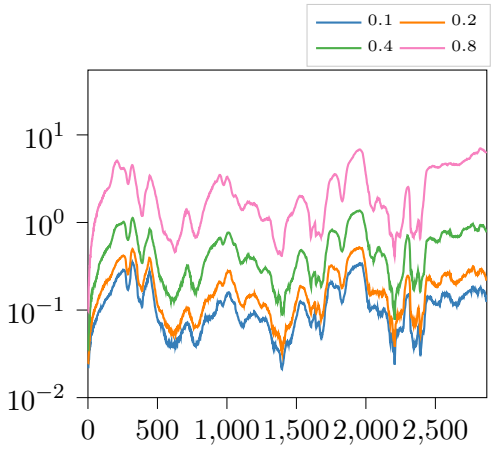
(a) Baseline [166] with varying $|\mathcal{J}|$



(b) Proposed with varying σ

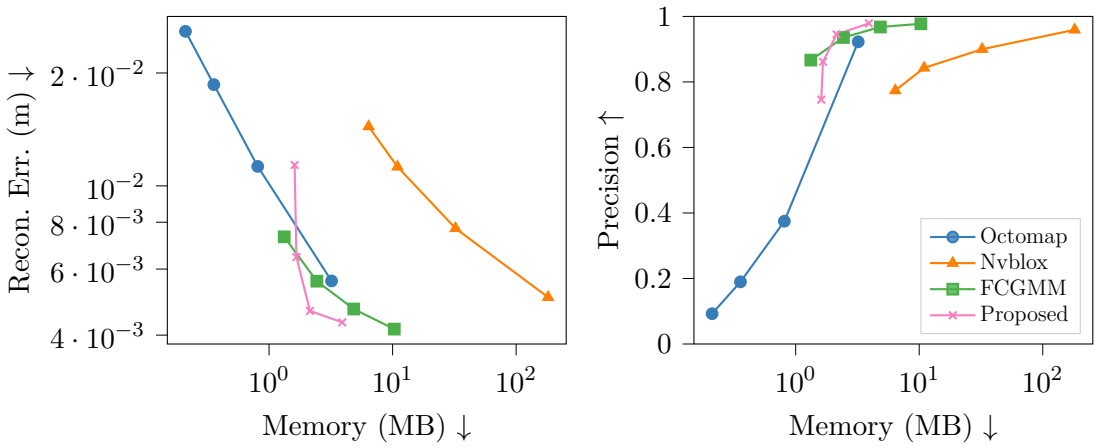
Method	Time (s)	Δ
FCGMM-800	44751.6	–
FCGMM-400	20702.3	–
FCGMM-200	9957.6	–
FCGMM-100	4774.7	–
Proposed-0.02	1239.6	**36.1**×
Proposed-0.03	666.1	**31.0**×
Proposed-0.04	488.0	**20.4**×
Proposed-0.05	436.9	**10.9**×

(c) Cumulative time taken



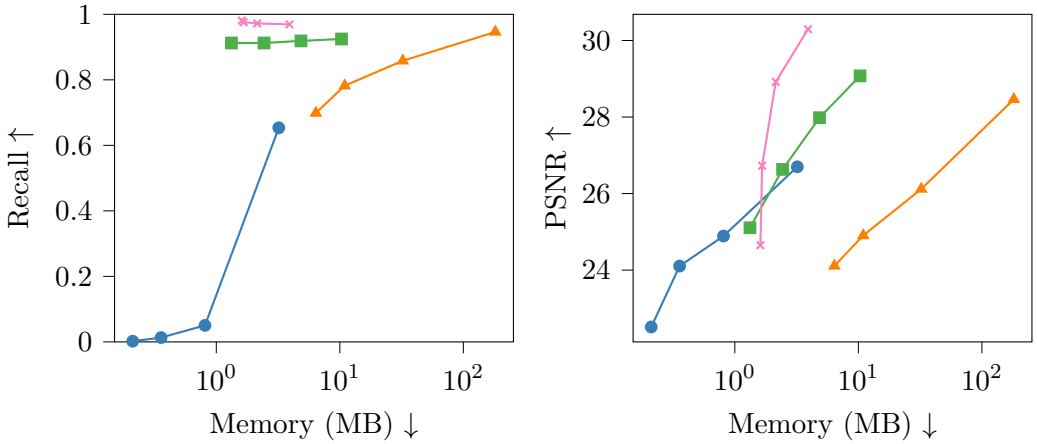
(d) Ablation for α

Figure 6.7: Comparison of the relevant subset $\mathcal{Z}^r$ calculation time between the prior work on multimodal GMM mapping [166] and the proposed approach. The per-frame calculation time in seconds is plotted for (a) different values of fixed numbers of components $|\mathcal{J}|$ and (b) different values of the bandwidth parameter σ for the proposed method. (c) Notice that the spatial hash (Sect. 6.1.4.3) enables an order of magnitude improvement and that the performance gains increase monotonically with model size. (d) shows an ablation of calculation times for different values of the spatial hash resolution parameter α.



(a) MRE (lower is better)

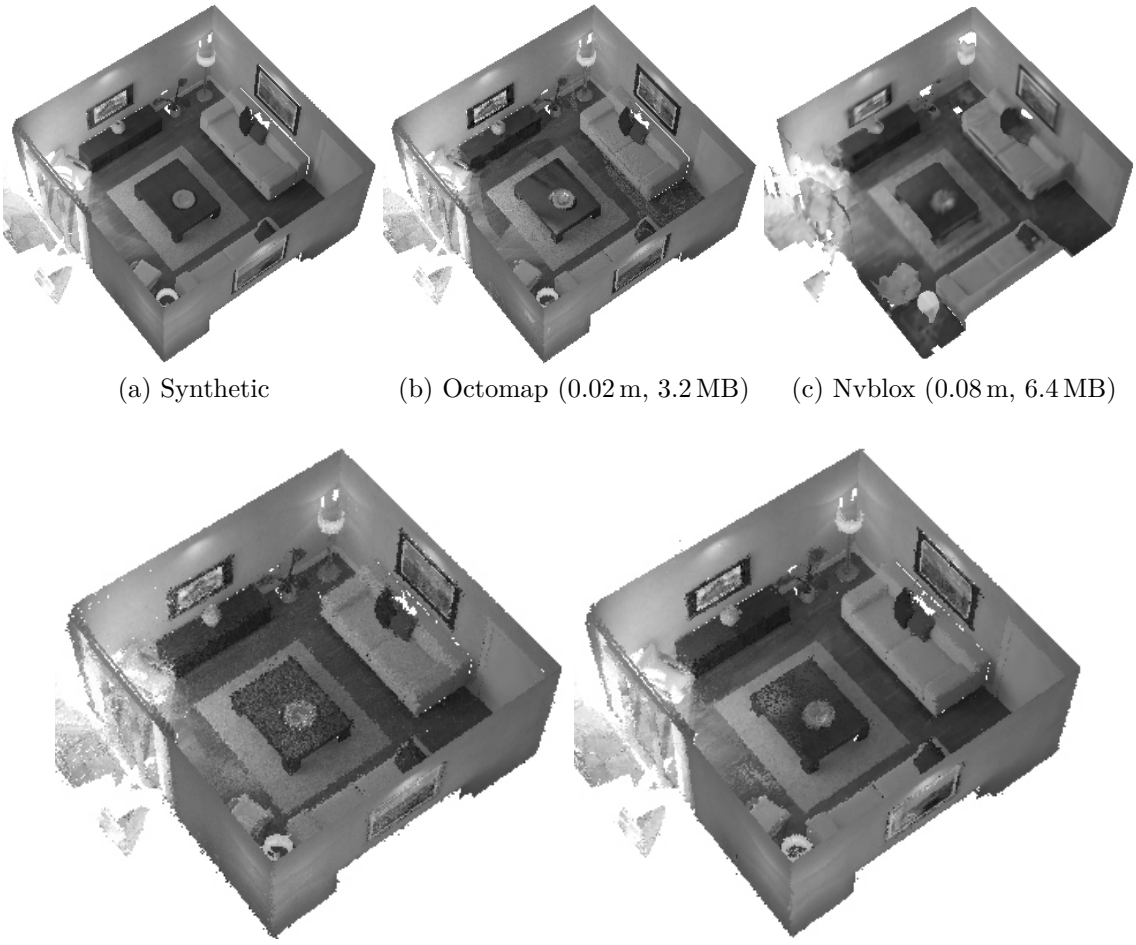
(b) Precision (higher is better)



(c) Recall (higher is better)

(d) PSNR (higher is better)

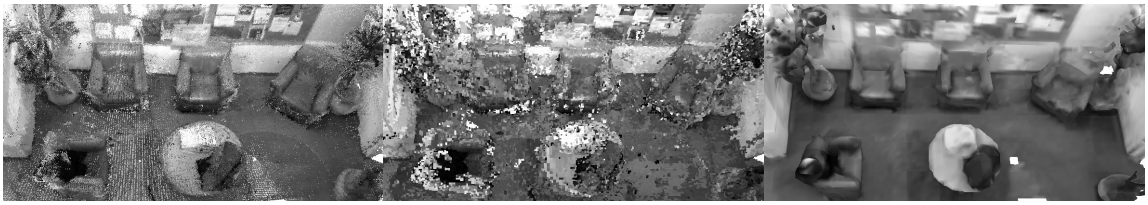
Figure 6.8: Quantitative comparison of (a) reconstruction error, (b) precision, (c) recall, and (d) PSNR as a function of the map size in megabytes (MB) for each approach. The dataset under consideration is the synthetic D1 dataset shown in Fig. 6.9a. Note that the proposed approach yields a map that requires less disk space than the competing methods while demonstrating at par or better reconstruction accuracy (i.e., low reconstruction error and high precision).



(a) Synthetic (b) Octomap (0.02 m, 3.2 MB) (c) Nvblox (0.08 m, 6.4 MB)

(d) FCGMM (400, 4.8 MB) (e) Proposed (0.02, 2.1 MB)

Figure 6.9: Qualitative comparison of the reconstructions obtained by baseline methods and the proposed approach at similar values of map size for (a). The highest achievable resolution used during execution and resulting map sizes are reported in the sub-captions. (b) visualizes the lowest level of the Octomap octree. Incorrect intensity values are visible due to the color averaging within the octree. (c) illustrates the mesh extracted from the stored TSDF for Nvblox. Aliasing is visible in the meshes due to large voxel sizes required for a lower memory footprint. (d) FCGMM and (e) the proposed method enable qualitatively similar high-resolution dense reconstructions; however, the FCGMM output requires a much longer time to process incremental observations (see Fig. 6.7). A video of the proposed approach reconstructing the D1 dataset is available at https://youtu.be/VgPEEcbUAnY.



(a) Real-World (b) Octomap (0.02 m, 5.9 MB) (c) Nvblox (0.06 m, 8.2 MB)

(d) FCGMM (400, 6.2 MB) (e) Proposed (0.02, 5.7 MB)

Figure 6.10: Same results as in Fig. 6.9 but with real-world point cloud data.

3060 GPU (12 GB RAM). The CPU implementation of the Octomap approach for color data is used. Nvblox and the proposed method[4] use the CPU and GPU for incremental mapping. Nvblox uses both depth and color data by default. It is modified to use depth and grayscale images for the comparison presented in this section. Since the software for prior GMM map works [166, 167, 180] is not openly available, the codebase for the proposed approach is modified to use a fixed number of components for the FCGMM comparison. The FCGMM approach uses the GPU for EM execution but CPU for the $\mathcal{Z}^r$ calculation because it requires access to more RAM than is available to the GPU.

6.2.3 Relevant Point Cloud Calculation

As mentioned in Section 6.1.4.1, a drawback of prior work [166] is that the relevant point cloud subset $\mathcal{Z}^r$ calculation in Eq. (6.8) is not real-time viable, especially as the number of components $|\mathcal{K}|$ increase in the global point cloud model $\mathcal{G}_G$. The proposed spatial hashing approach reduces this computation cost by selecting a subset of components $\mathcal{B}$ that geometrically overlaps the point cloud. Figure 6.7 demonstrates the performance gains due to the proposed approach. While the methodology proposed in [166] is hierarchical, we choose to compare against their highest fidelity model (i.e., lowest layer) in the hierarchy for a fair comparison.

For the baseline approach (Fig. 6.7a), the calculation times per frame are shown for increasing values of number of mixture components $|\mathcal{J}|$, $\boldsymbol{J} = \{100, 200, 400, 800\}$. The proposed methodology enables an order of magnitude faster $\mathcal{Z}^r$ calculation as compared to the baseline approach, because the log-likelihood operates over all mixture components and points. The spatial hash resolution α is fixed at 0.2 m for all

[4]Release `0.1.0` of https://github.com/gira3d/gira3d-reconstruction.

Method	Param.	MRE (m) ↓	Prec. ↑	Rec. ↑	PSNR (dB) ↑	Mem. (MB) ↓
Octomap	0.02	0.007	0.81	0.46	18.37	5.94
	0.04	0.011	0.55	0.06	16.76	1.13
	0.06	0.016	0.41	0.02	15.84	0.44
	0.08	**0.021**	**0.33**	**0.01**	**15.30**	**0.23**
Nvblox	0.02	0.006	0.92	0.38	20.63	**114.19**
	0.04	0.008	0.86	0.35	19.08	19.98
	0.06	0.011	0.81	0.31	18.26	8.19
	0.08	0.014	0.76	0.28	17.51	4.22
FCGMM	800	**0.005**	**0.98**	**0.72**	20.25	14.11
	400	0.006	0.97	**0.72**	19.35	6.24
	200	**0.005**	0.96	**0.72**	18.64	2.98
	100	0.006	0.92	**0.72**	17.62	1.46
Proposed	0.02	**0.005**	**0.98**	0.65	**20.74**	5.71
	0.03	**0.005**	0.97	0.64	20.18	2.39
	0.04	0.006	0.96	0.65	19.76	1.32
	0.05	0.006	0.93	0.65	19.12	0.83

(a) D2: *Lounge* dataset

Method	Param.	MRE (m) ↓	Prec. ↑	Rec. ↑	PSNR (dB) ↑	Mem. (MB) ↓
Octomap	0.02	0.007	0.83	0.51	18.61	5.93
	0.04	0.011	0.57	0.06	17.35	1.10
	0.06	0.016	0.44	0.02	16.49	0.43
	0.08	**0.021**	**0.36**	**0.01**	**15.95**	**0.23**
Nvblox	0.02	0.006	0.92	0.38	20.94	**96.95**
	0.04	0.009	0.87	0.33	19.71	18.61
	0.06	0.010	0.83	0.29	18.86	8.00
	0.08	0.010	0.81	0.26	18.29	4.40
FCGMM	800	–	–	–	–	–
	400	**0.004**	0.97	**0.83**	20.66	10.68
	200	0.005	0.95	**0.83**	19.70	4.85
	100	0.007	0.91	**0.83**	18.59	2.30
Proposed	0.02	0.005	**0.98**	0.70	**21.42**	5.59
	0.03	0.005	0.97	0.68	20.96	2.48
	0.04	0.005	0.95	0.66	20.56	1.39
	0.05	0.006	0.93	0.66	20.05	0.88

(b) D3: *Copyroom* dataset

Method	Param.	MRE (m) ↓	Prec. ↑	Rec. ↑	PSNR (dB) ↑	Mem. (MB) ↓
Octomap	0.02	0.007	0.84	0.62	19.19	5.29
	0.04	0.011	0.59	0.08	18.32	0.95
	0.06	0.016	0.47	0.02	17.61	0.37
	0.08	**0.021**	**0.39**	**0.01**	**17.79**	**0.19**
Nvblox	0.02	0.005	**0.99**	0.42	**24.80**	**155.05**
	0.04	0.005	0.98	0.39	23.41	25.96
	0.06	0.005	0.96	0.37	22.40	9.43
	0.08	0.006	0.94	0.34	21.43	4.89
FCGMM	800	0.005	**0.99**	**0.82**	22.13	13.34
	400	**0.004**	**0.99**	0.80	21.51	5.66
	200	0.005	0.98	0.79	20.99	2.60
	100	0.006	0.96	0.79	20.40	1.24
Proposed	0.02	0.005	**0.99**	0.68	21.83	3.27
	0.03	0.005	0.98	0.66	21.62	1.26
	0.04	0.005	0.97	0.65	21.41	0.63
	0.05	0.006	0.95	0.65	21.16	0.36

(c) D4: *Stonewall* dataset

Figure 6.11: Quantitative comparison of Octomap, Nvblox, FCGMM, and the proposed approach using the real-world datasets with noisy RGB-D data. The best and worst values in each column are colored green and red respectively. The FCGMM method results in a larger map size compared to the proposed approach and is orders of magnitude slower in execution time (Fig. 6.7). These results highlight that the proposed approach balances the accuracy and map size better than the state-of-the-art approaches.

values of σ. Because lower σ yields higher resolution reconstruction, the computation time increases as σ decreases. Figure 6.7c presents the cumulative $\mathcal{Z}^r$ calculation times for the D1 dataset. The Δ columns show the order of magnitude improvement via the proposed approach. Notice that while the performance gain of Proposed-0.05 compared to FCGMM-100 is nearly $\mathbf{10}\times$, it increases with the model fidelity; the Proposed-0.02 is about $\mathbf{30}\times$ faster than FCGMM-800. Finally, Fig. 6.7d compares the $\mathcal{Z}^r$ calculation times for $\alpha = \{0.1, 0.2, 0.4, 0.8\}$. Increasing α results in overall higher calculation times because the size of $\mathcal{B}$ gets larger. Before using the proposed method, the value of α should be set according to the available CPU computation resources.

6.2.4 Global Map Accuracy and Compression

For Octomap and Nvblox, the predicted point cloud $\mathcal{Z}^{\text{pr}}$ and mesh are constructed, respectively, after processing all the frames in a given dataset. For FCGMM and the proposed approach, the predicted point cloud is inferred from the global model $\mathcal{G}_G$ using the method in Sect. 6.1.5. The Octomap method requires specifying a minimum leaf size for the underlying octree used for modeling and inference. We use a range of leaf sizes for the experiments, $\boldsymbol{\alpha}_{\text{om}} = \{0.02, 0.04, 0.06, 0.08\}$m. The same set of values are used for the voxel sizes required in the Nvblox method, $\boldsymbol{\alpha}_{\text{nv}} = \boldsymbol{\alpha}_{\text{om}}$. For FCGMM and the proposed approach, the same set of parameters are used as in Sect. 6.2.3. The ground truth point cloud, $\mathcal{Z}^{\text{gt}}$, is constructed by appending all the point clouds corresponding to the images and poses in the dataset followed by downsampling using a voxel grid filter with a small voxel size (0.01 m for all experiments in this section).

The performance measures for 3D reconstruction are (1) Mean Reconstruction Error (MRE), which is the average distance of the closest points between $\mathcal{Z}^{\text{pr}}$ and $\mathcal{Z}^{\text{gt}}$ (lower is better), (2) Precision of 3D reconstruction, which measures the fraction of points in $\mathcal{Z}^{\text{pr}}$ that lie within 0.01 m of a point in $\mathcal{Z}^{\text{gt}}$ (higher is better), and (3) Recall of 3D reconstruction, which measures the fraction of points in $\mathcal{Z}^{\text{gt}}$ that lie within 0.01 m of a point in $\mathcal{Z}^{\text{pr}}$ (higher is better). Intuitively, this measure computes the degree of "completeness" in the reconstruction. The performance measure for intensity reconstruction is the peak-signal-to-noise ratio (PSNR) calculated using the mean squared error (MSE) between the intensity values of the closest points in $\mathcal{Z}^{\text{pr}}$ and $\mathcal{Z}^{\text{gt}}$ (higher PSNR is better). These measures are computed using the closest point distance computation functions for point clouds in Open3D [211]. For the Nvblox case in particular, the mesh output is uniformly and densely sampled to create a point cloud with number of points equal to the ground truth point cloud.

The memory storage efficiency of the multimodal environment representations is measured by calculating the size (measured in bytes) of the models that can be loaded from disk to create $\mathcal{Z}^{\text{pr}}$. The models are chosen so that they can enable reconstruction of the surface and occupancy modeling for other robots in a multi-robot reconstruction scenario. For Octomap, the model size corresponds to the output `.ot` file [88], as opposed to the binary `.bt` file that does not retain occupancy information. Due to the same reason, for Nvblox the SQLite3 database (`.db` file) output is used instead of the output `.ply` mesh file. For the FCGMM and proposed methodologies, the

memory occupied by the means, covariances, and weights in the global model $\mathcal{G}_G$ is calculated (four floats for each mean, one float for each weight, and ten floats for each covariance). Note that occupancy modeling from a stored GMM map with these parameters has been demonstrated in prior work [61, 167, 138, 180].

Figure 6.8 shows the variation in the performance measures for all methods with respect to the map size on disk for the D1 dataset. Each data point in the plots corresponds to a unique parameter setting in $\boldsymbol{\alpha}_{\text{om}}$, $\boldsymbol{\alpha}_{\text{nv}}$, $\boldsymbol{J}$, and $\boldsymbol{\sigma}$ for Octomap, Nvblox, FCGMM, and the proposed approach respectively. To attain similar levels of mean reconstruction error and precision, the proposed approach, FCGMM, and Octomap require an order of magnitude less memory than Nvblox (Figs. 6.8a and 6.8b). This is because Nvblox utilizes a regular grid of fixed resolution and multiple data storage layers while Octomap, FCGMM, and the proposed approach leverage octrees, GMMs, and SOGMMs, respectively. Note that while these values are close for the FCGMM and the proposed methods, in the FCGMM case the time taken to create the model is much higher (Sect. 6.2.3).

The FCGMM and the proposed approaches achieve a recall score close to 1.0 (Fig. 6.8c) demonstrating that for nearly each point in the ground truth point cloud, there is a point in the reconstruction within a 0.01 m ball. Both of these GMM-based methods outperform Octomap because an arbitrarily high number of points can be densely sampled from a GMM (Sect. 6.1.5) whereas the Octomap outputs the point cloud at its minimum pre-specified leaf size. The Nvblox method output mesh is uniformly sampled; however, a low voxel size is required to achieve similar recall scores.

The highest intensity reconstruction accuracy (i.e., PSNR score in Fig. 6.8d) attained by Octomap (PSNR = 26.70) is much lower than the proposed approach (PSNR = 30.29) at a similar storage cost. This is because the intensity in an Octomap octree node is averaged according to the density of the points around the node. In contrast, the proposed approach treats intensity as a univariate random variable and jointly modeled with the 3D coordinates in the global point cloud model. Inference from this joint probability density leads to a higher accuracy than the averaging in Octomap. Nvblox fuses intensity information into the 3D map using a weighted average update. This process is an improvement over Octomap but still requires a low voxel size to attain a PSNR comparable to GMM-based approaches. Finally, the FCGMM approach demonstrates a lower PSNR than the proposed approach for similar storage costs. This is because the FCGMM uses a fixed number of components for every scene in the dataset while the proposed method uses SOGMM, which adapts the number of components according to the complexity of depth and image data [76]. The impact of these quantitative results is visible in the qualitative comparison shown in Fig. 6.9. The reconstructions from all methods are shown for comparable map sizes along with the ground truth point cloud for D1 and D2 datasets.

Figure 6.11 provides performance statistics corresponding to real-world datasets D2, D3 and D4, which exhibit noisy sensor readings. For each performance measure, the best and worst values are highlighted in green and red, respectively. Note that there is no result for the D2 dataset in the FCGMM-800 case because the relevant point cloud calculation required more RAM than the available 32 GB. This is expected

since the D2 dataset contains 5490 frames which is nearly twice the other datasets. While using Octomap at a 0.08 m resolution results in the lowest map size (0.23 MB), reconstruction performance is poor compared to all other methods. Nvblox outputs the largest map size on disk at 114.19 MB but does not enable the highest PSNR. The FCGMM and proposed approach enable similar reconstruction accuracy; however, the proposed approach results in smaller map sizes and utilizes less computation as shown earlier. This trend is observed for the D2 and D3 datasets as well. One exception for Nvblox is that it provides a higher PSNR compared to the proposed approach for the D3 dataset but it consumes about 50× more storage.

6.3 Summary

In this chapter, a continuous probabilistic modeling approach is presented towards addressing the challenge **C3**. We enable adaptation in this framework through estimation of number of components using the principle of relevant information from information-theoretic learning. The quantitative and qualitative results for the proposed method demonstrate its efficacy in terms of reconstruction accuracy and memory utilization on diverse real-world scenes. For streaming sensor data, inserting a new point cloud observation to an existing GMM map model involves iterating over all the mixture components; which is computationally expensive. To bridge this gaps, this chapter formulated methodologies to (1) extract a submap by innovating a spatial hash table of mixture components and (2) incrementally update the global environment model in a computationally efficient manner. The approach was evaluated with synthetic and real-world datasets and the results demonstrated that the proposed approach enables high-fidelity reconstruction at low memory with an order of magnitude increase in speed compared to existing GMM-based mapping methods.

The implementation details for these methods for CPU and GPU platforms are provided in Appendix A.

CHAPTER 7

Collision Avoidance using Self-Organizing GMMs

This chapter proposes a method to compute distance at any point in space for an ellipsoidal robot from a surface represented by a set of Gaussians.

Given an ellipsoidal robot body model and a set of Gaussians representing the surface point cloud, this chapter contributes methods to compute the continuous-space

1. Euclidean distance estimate between the robot body and surface,
2. the approximate gradient of this distance, and
3. the upper bound on collision probability when the robot position is a Gaussian random variable.

These methods are evaluated using 2D and 3D surface point clouds from simulation and real-world environments. Several discrete- and continuous-space methodologies have been proposed to compute Euclidean Distance Fields (EDFs), which indicates distance from an obstacle using only positive values, and Euclidean Signed Distance Fields (ESDFs), which indicate occluded distance using a negative sign. While the applications of these quantities are quite diverse (e.g., shape reconstruction, meshing, etc.), we review continuous-space methods that have been proposed for robot navigation purposes. Collision probability calculation methods under robot position uncertainty are also discussed.

This capability is important to enable collision avoidance from SOGMM (Challenge **C4**). Section 7.1 details the proposed methods. The evaluation is presented in Sect. 7.2. A summary is provided in Sect. 7.3.

7.1 Approach

Starting with the preliminary information Sect. 7.1.1, the problem statement is provided in Sect. 7.1.2. The proposed methods are detailed in Sect. 7.1.3 and Sect. 7.1.4. In this section, small letters are scalars (e.g. x, y), bolded small letters are vectors (e.g. $\mathbf{x}$, $\mathbf{y}$), capital letters are random variables (e.g. X, Y), capital bolded letters are matrices (e.g. $\mathbf{X}$, $\mathbf{Y}$), and calligraphic letters are sets (e.g. $\mathcal{X}, \mathcal{Y}$).

7.1.1 Preliminaries

A solid ellipsoid of dimension q, center $\mathbf{p} \in \mathbb{R}^q$, and shape $\mathbf{P} \in \mathbb{R}^{q \times q}$ is the quadratic form inequality

$$\mathcal{E}(\mathbf{p}, \mathbf{P}) = \{\mathbf{q} \in \mathbb{R}^q \mid (\mathbf{q} - \mathbf{p})^\top \mathbf{P} (\mathbf{q} - \mathbf{p}) \leq 1\}.$$

The eigen decomposition of $\mathbf{P}$ enables determining the rotation and scale of the ellipsoid. If the eigen decomposition of $\mathbf{P}$ is given by $\mathbf{P} = \mathbf{R}\mathbf{S}\mathbf{R}^\top$, then $\mathbf{R}$ is an orthogonal matrix that provides the ellipsoid's principal axes rotation and $\mathbf{S}$ is a diagonal matrix where entries are the inverse squares of the semi-principal axes lengths. The positive definite matrix $\mathbf{P}^{-1}$ is called the shape matrix of the ellipsoid [80].

The region inside and on a probability isocontour of the probability density function for a q-variate Gaussian random variable X with mean vector $\boldsymbol{\mu}_X$ and covariance matrix $\boldsymbol{\Sigma}_X$ can be geometrically interpreted as an ellipsoid [14, Ch. 2]. The l-th level isocontour is given by the solid ellipsoid $\mathcal{E}_X(\boldsymbol{\mu}_X, \mathbf{R}\boldsymbol{\Gamma}\mathbf{R}^\top)$. The rotation matrix $\mathbf{R}$ contains the eigenvectors of $\boldsymbol{\Sigma}_X^{-1}$ as columns. The entries of the diagonal matrix $\boldsymbol{\Gamma}$ are given by $1/l^2$ times the eigenvalues of $\boldsymbol{\Sigma}_X^{-1}$. For reference, $l = 3$ and 4 provide 99.7% and 99.95% coverage bounds respectively on the points modeled by the random variable X.

If the probability density function is instead given by a weighted sum of Gaussian density functions (i.e., a GMM), then the probability isocontours can be approximated as a set of ellipsoids with one ellipsoid per component of the mixture. Note that this geometric interpretation ignores the weights of the GMM.

7.1.2 Problem Statement

A robot $\mathcal{P}$ is equipped with a range sensor (e.g. depth camera, LiDAR, etc.) and it can be moved in a workspace $\mathcal{W} \subseteq \mathbb{R}^q$. In this work q is either 2 or 3. The region inside the workspace occupied by the robot is modeled using the solid ellipsoid $\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P})$. The parameters $\mathbf{p}$ and $\mathbf{P}$ are determined using the Lowner-John ellipsoid fit [157]. The center $\mathbf{p}$ can be uncertain. This uncertainty is modeled using a continuous multivariate Gaussian variable P with the probability density function $\mathcal{N}(\mathbf{p}; \boldsymbol{\mu}_P, \boldsymbol{\Sigma}_P)$. In practice, these uncertainty estimates may be provided by an external state estimation system [129].

The onboard range sensor provides a stream of point cloud measurements $\mathcal{Z}$ of the surfaces $\mathcal{W}_\text{s}$ in the workspace. The elements of the point cloud $\mathcal{Z}$ are assumed to be independent and identically distributed samples of a random variable Z with the probability density function given by the GMM

$$f_Z = \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_Z^m, \boldsymbol{\Sigma}_Z^m) \tag{7.1}$$

where π_m, $\boldsymbol{\mu}_Z^m$, and $\boldsymbol{\Sigma}_Z^m$ are the weight, mean, and covariance of the m-th Gaussian component in an M-component GMM. This GMM can be obtained from many recent point cloud modeling methods. The weights may be uniform, such as the GSMs

obtained via stochastic gradient-descent [99, 98] or geometric region growing [50]. Alternatively, GMMs with non-uniform weights obtained from Expectation Maximization [71, 167] or scan line segmentation [110] may also be used. The proposed methods in this work only depend on the geometric interpretation of the GMMs and thus do not require the GMMs to be created using a specific method.

Given this information, the following problems are addressed in this work:

1. Estimate the closest distance between $\mathcal{E}_{\mathcal{P}}(\mathbf{p}, \mathbf{P})$ and f_Z. Estimate the gradient of this distance at any location. The solution must not depend on a discretization of the space (Sect. 7.1.3).

2. Estimate the collision probability between $\mathcal{E}_{\mathcal{P}}(\mathbf{p}, \mathbf{P})$ and f_Z when the ellipsoid center position is uncertain (Sect. 7.1.4).

7.1.3 Continuous Euclidean Distance Field

The m-th component of the GMM density f_Z from Eq. (7.1) can be interpreted as an ellipsoid (Sect. 7.1.1). If it were possible to efficiently calculate the distance of the robot to all M components, we could compute the minimum distance of the robot to the surface GMM. Thus, we require a method to find the closest distance between two ellipsoids. Unfortunately, this distance cannot be derived as a closed-form expression [112].

Many optimization-based approaches have been proposed for this purpose [112, 157, 187]. In this work, the method by Rimon and Boyd [157] is leveraged because it formulates the optimization as an eigenvalue problem. However, the expressions stated in their work require several matrix inversions and square roots. It is not obvious how to avoid these calculations and leverage numerically stable and faster alternatives (e.g. linear system solvers, decompositions) without introducing approximations. The following proposition restates the result from [157] but removes the need to explicitly calculate matrix inversions and square roots.

Proposition 1 (Deterministic Ellipsoid Distance [157][1])**.** Consider two ellipsoids $\mathcal{E}_1(\mathbf{b}, \mathbf{B})$ and $\mathcal{E}_2(\mathbf{c}, \mathbf{C})$. The centers and shapes of these ellipsoids are perfectly known. Let $\mathbf{B} = \hat{\mathbf{B}}\mathbf{\Lambda}_B\hat{\mathbf{B}}^\top$ and $\mathbf{C} = \hat{\mathbf{C}}\mathbf{\Lambda}_C\hat{\mathbf{C}}^\top$ be the eigen decompositions of $\mathbf{B}$ and $\mathbf{C}$ respectively. Consequently, $\mathbf{B}^{1/2} = \hat{\mathbf{B}}\mathbf{\Lambda}_B^{1/2}\hat{\mathbf{B}}^\top$, $\mathbf{B}^{-1} = \hat{\mathbf{B}}\mathbf{\Lambda}_B^{-1}\hat{\mathbf{B}}^\top$, $\mathbf{B}^{-1/2} = \hat{\mathbf{B}}\mathbf{\Lambda}_B^{-1/2}\hat{\mathbf{B}}^\top$, and $\mathbf{C}^{-1} = \hat{\mathbf{C}}\mathbf{\Lambda}_C^{-1}\hat{\mathbf{C}}^\top$. Let λ be the minimal eigenvalue[2] of the $2q \times 2q$ matrix

$$\mathbf{M}_1 = \begin{bmatrix} \tilde{\mathbf{C}} & -\mathbf{I}_q \\ -\tilde{\mathbf{c}}\tilde{\mathbf{c}}^\top & \tilde{\mathbf{C}} \end{bmatrix}$$

such that $\mathbf{I}_q$ is an identity matrix of order q, $\tilde{\mathbf{C}} = \mathbf{B}^{1/2}\mathbf{C}^{-1}\mathbf{B}^{1/2}$, and $\tilde{\mathbf{c}}$ is the solution of the linear system

$$(\mathbf{B}^{-1/2}\hat{\mathbf{Q}}\mathbf{\Lambda}_Q^{1/2}\hat{\mathbf{Q}}^\top)\tilde{\mathbf{c}} = \mathbf{c} - \mathbf{b} \tag{7.2}$$

[1] The typographical errors in the expressions for $\tilde{\mathbf{b}}$ and $\mathbf{y}^*$ in [157, Prop. 3.2] have been corrected here.

[2] The minimal eigenvalue is equal to the real part of the eigenvalue with the lowest real part amongst all eigenvalues (real or complex). This notion is required because the eigenvalues of $\mathbf{M}_1$ and $\mathbf{M}_2$ may be complex numbers.

where $\hat{\mathbf{Q}}\mathbf{\Lambda}_Q\hat{\mathbf{Q}}^\top$ is the eigen decomposition of the real symmetric matrix $\mathbf{B}^{-1/2}\mathbf{C}\mathbf{B}^{-1/2}$. Let μ be the minimal eigenvalue of the $2q \times 2q$ matrix

$$\mathbf{M}_2 = \begin{bmatrix} \mathbf{B}^{-1} & -\mathbf{I}_q \\ -\tilde{\mathbf{b}}\tilde{\mathbf{b}}^\top & \mathbf{B}^{-1} \end{bmatrix}$$

such that $\tilde{\mathbf{b}} = -\lambda\mathbf{B}^{-1/2}\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}$ is the solution to the linear system

$$\{\mathbf{B}^{-1/2}(\lambda\mathbf{I}_q - \tilde{\mathbf{C}})\mathbf{B}^{1/2}\}\boldsymbol{\alpha} = \mathbf{c} - \mathbf{b}. \tag{7.3}$$

Given these quantities, the closest distance estimate $d(\mathcal{E}_1, \mathcal{E}_2)$ between $\mathcal{E}_1$ and $\mathcal{E}_2$ is

$$d(\mathcal{E}_1, \mathcal{E}_2) = \|\mathbf{d}^*\|, \tag{7.4}$$

where $\|.\|$ denotes the L2-norm of a vector and $\mathbf{d}^*$ is the solution to the linear system

$$(\mu\mathbf{I}_q - \mathbf{B}^{-1})\mathbf{d}^* = -\mu\lambda\boldsymbol{\alpha}. \tag{7.5}$$

Note that calculating matrix inverses and square roots of diagonal matrices like $\mathbf{\Lambda}_B$ only requires inverse and square root operations on their scalar diagonal entries as opposed to full matrix operations. Therefore, there are no explicit matrix inversion or matrix square root calculations required in Proposition 1.

Using Proposition 1, the distance between the robot body ellipsoid $\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P})$ and the surface model f_Z can be computed by computing the minimum over all M components of f_Z:

$$d^*(\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P}), f_Z) = \min_m d(\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P}), \mathcal{E}_m) \tag{7.6}$$

where $d(.)$ is the distance function from Eq. (7.4) and $\mathcal{E}_m$ denotes the ellipsoid corresponding to the m-th Gaussian component in the GMM density f_Z. This ellipsoid can be constructed for isocontours of the Gaussian component Sect. 7.1.1.

To calculate the gradient, Rimon and Boyd [157] suggest deriving the analytical gradient of Eq. (7.4) as it is differentiable. However, computing this gradient requires several matrix multiplications and inversions. To save computational resources onboard robots, an approximation is leveraged. From Eq. (7.6), we also get the ellipsoid on the surface, $\mathcal{E}_m^*$, that is closest to the robot. The gradient vector is approximated using the position vector $\mathbf{d}^*$ (from Eq. (7.4)) for $\mathcal{E}_m^*$ and $\mathcal{E}_\mathcal{P}$,

$$\nabla d^*(\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P}), f_Z) = \frac{\mathbf{d}^*}{\sqrt{\mathbf{d}^{*\top}\mathbf{d}^*}}. \tag{7.7}$$

Under the limiting condition where the number of components M is equal to the number of points in the point cloud $\mathcal{Z}$ such that each point in the point cloud is represented with a Gaussian component, the distance (Eq. (7.6)) and gradient (Eq. (7.7)) formulations are exact (i.e., ground truth values). As M decreases, the error in d^* and ∇d^* increases relative to the ground truth value.

The computation in Eq. (7.6) scales linearly with the number of components M in the GMM. Local submap extraction approaches such as hash maps [71] or spatial partitioning data structures such as KD-tree [90] and B+-tree [130] may enable improved scalability as M increases.

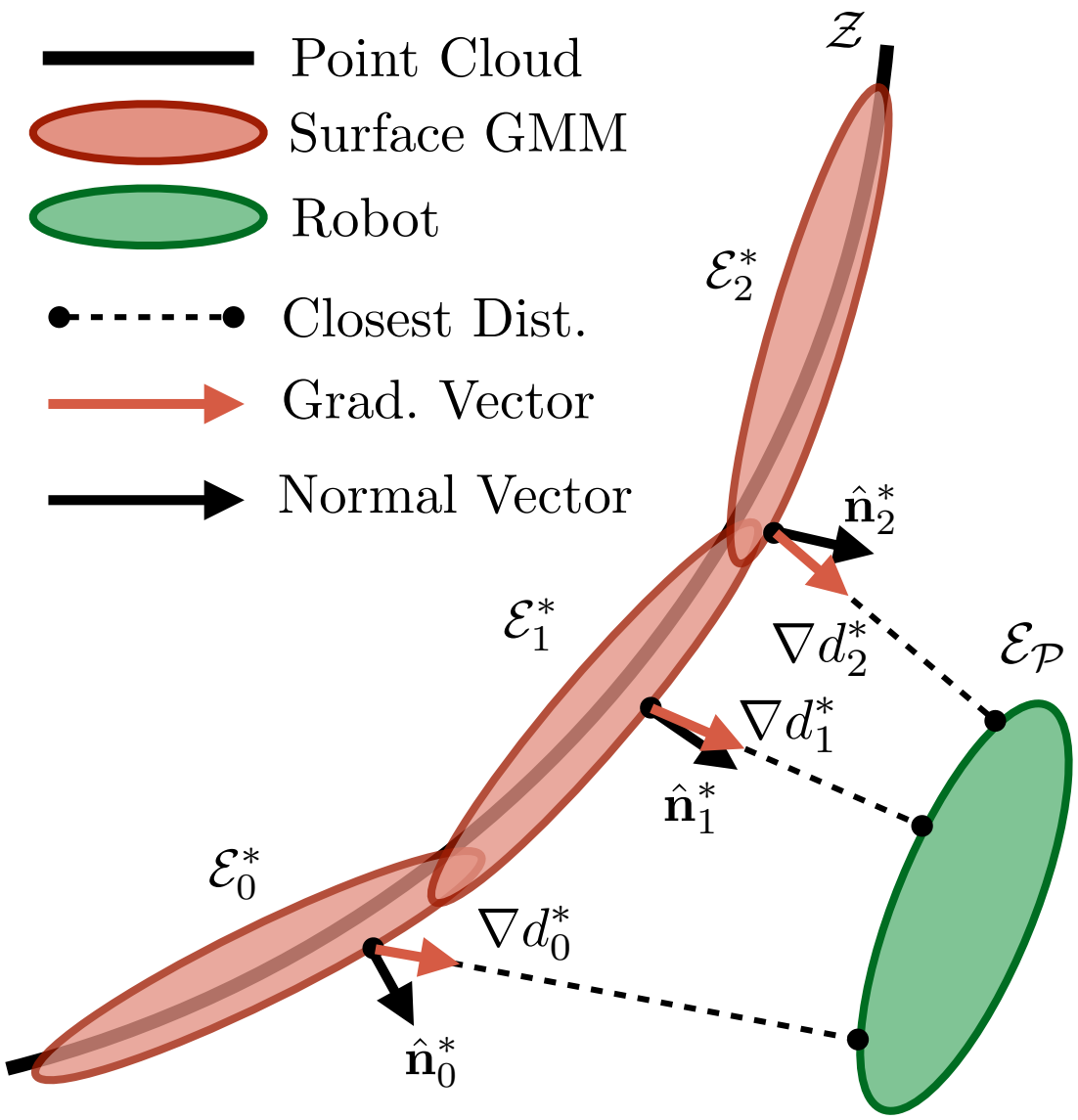


Figure 7.1: Illustration of the quantities required for blending weights calculation in collision probability estimation (Sect. 7.1.4). For each of the ellipsoids on the surface, the blending weight w_k is the dot product of the distance gradient ∇d_k^* and the normal $\hat{\mathbf{n}}_k^*$. *This figure is best viewed in color.*

7.1.4 Collision Probability Under Position Uncertainty

The following corollary to Proposition 1 will be used in this section.

Corollary 1 (Deterministic Ellipsoid Collision Check [183]). If it is desired to only check whether $\mathcal{E}_1$ and $\mathcal{E}_2$ collide (under deterministic conditions), an explicit calculation of Eq. (7.4) is not required. When $\mathcal{E}_1$ and $\mathcal{E}_2$ touch or intersect, Thomas et al. [183] show that $\mathbf{y}^\top\mathbf{D}^\top\mathbf{B}\mathbf{D}\mathbf{y} \leq 1/\lambda^2$, where $\mathbf{y} = \mathbf{c} - \mathbf{b}$, $\mathbf{D} = \mathbf{B}^{-1/2}(\lambda\mathbf{I}_q - \tilde{\mathbf{C}})^{-1}\mathbf{B}^{1/2}$, and other quantities are as calculated in Proposition 1. We simplify this inequality to

$$\mathbf{y}^\top\mathbf{B}^{1/2}\mathbf{A}^{-1}\mathbf{B}^{1/2}\mathbf{y} \leq 1/\lambda^2 \tag{7.8}$$

where $\mathbf{A} = (\lambda\mathbf{I}_q - \tilde{\mathbf{C}})^2$ is a real symmetric matrix. It is shown in [157, Thm. 2] that when the center of ellipsoid $\mathcal{E}_1$, $\mathbf{b}$, lies outside of $\mathcal{E}_2$[3], λ is always negative. Since $\tilde{\mathbf{C}} = \mathbf{B}^{1/2}\mathbf{C}^{-1}\mathbf{B}^{1/2}$ with $\mathbf{B} \succ 0$ and $\mathbf{C} \succ 0$, it follows that $\tilde{\mathbf{C}} \succ 0$. Therefore, the matrix $\lambda\mathbf{I}_q - \tilde{\mathbf{C}} \prec 0$, which implies $\mathbf{A} \succ 0$ (i.e., $\mathbf{A}$ is positive definite). Therefore, the Cholesky decomposition of $\mathbf{A}$ can be used to efficiently compute Eq. (7.8) without explicit matrix inversions or square roots.

Let $\mathbf{v} = \mathbf{y}^\top\bar{\mathbf{A}}\mathbf{y}$ where $\bar{\mathbf{A}} = \mathbf{B}^{1/2}\mathbf{A}^{-1}\mathbf{B}^{1/2}$. When the center $\mathbf{b}$ of the ellipsoid $\mathcal{E}_1$ is a Gaussian distributed random variable with density $\mathcal{N}(\mathbf{b}; \boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B)$, $\mathbf{v}$ becomes a random variable as both $\mathbf{y}$ and $\bar{\mathbf{A}}$ depend on $\mathbf{b}$. Under a conservative assumption that $\bar{\mathbf{A}}$ and λ are deterministic (calculated using the mean $\boldsymbol{\mu}_B$)[4], it is proved in [183] that $P_B(\mathcal{E}_1, \mathcal{E}_2) \equiv P_B(\mathbf{v} \leq 1/\lambda^2)$ and an upper bound on the probability can be estimated via

$$P_B(\mathbf{v} \leq 1/\lambda^2) \leq \frac{\eta\sqrt{\mathbb{V}[\mathbf{v}]}}{\mathbb{E}[\mathbf{v}] + \eta\sqrt{\mathbb{V}[\mathbf{v}]} - (1/\lambda^2)} \tag{7.9}$$

where $\mathbb{E}[\mathbf{v}]$ and $\mathbb{V}[\mathbf{v}]$ denote the expectation and variance of $\mathbf{v}$, λ is as defined in Proposition 1, and η is a constant. The values $\mathbb{E}[\mathbf{v}]$ and $\mathbb{V}[\mathbf{v}]$ can be exactly calculated using [121, Thm. 3.2b.2]

$$\begin{aligned}\mathbb{E}[\mathbf{v}] &= \mathrm{tr}[\bar{\mathbf{A}}\boldsymbol{\Sigma}_B] + (\mathbf{c} - \boldsymbol{\mu}_B)^\top\bar{\mathbf{A}}(\mathbf{c} - \boldsymbol{\mu}_B) \text{ and} \\ \mathbb{V}[\mathbf{v}] &= 2\mathrm{tr}[(\bar{\mathbf{A}}\boldsymbol{\Sigma}_B)^2] + 4(\mathbf{c} - \boldsymbol{\mu}_B)^\top\bar{\mathbf{A}}\boldsymbol{\Sigma}_B\bar{\mathbf{A}}(\mathbf{c} - \boldsymbol{\mu}_B)\end{aligned}$$

where tr[.] denotes the trace of a matrix. Note that explicit matrix square roots are required in this upper bound calculation (Proposition 1). Estimation of the constant η is difficult. Thomas et al. [183] set this value to 0.25 for all cases[5]. However, we found that in some cases this value leads the denominator in Eq. (7.9) to be negative (i.e., $\mathbb{E}[\mathbf{v}] + \eta\sqrt{\mathbb{V}[\mathbf{v}]} < (1/\lambda^2)$). Therefore, in practice when this denominator is negative, starting with $\eta = 0.25$, we keep increasing it by 0.5 until the denominator turns positive. This is a valid approximation because η is used to upper bound the value of

[3]This assumption does not lead to a loss of generality because when $\mathbf{b}$ lies inside $\mathcal{E}_2$, the ellipsoids are definitely intersecting.

[4]Liu et al. [115] propose leveraging the Minkowski sum of ellipsoids to relax this assumption. However, it is computationally difficult to obtain a tight approximation of this sum [80].

[5]There is a typographical error in [183] which states that $\eta = 1$. The tight upper bound calculated in Section IV-D of [183] is correct when $\eta = 0.25$.

$\mathbf{v}$ [183]; increasing η makes this bound loose for certain ellipsoid pairs but maintains the validity of Eq. (7.9).

To compute the probability in Eq. (7.9) relative to the surface GMM f_Z, one approach is to identify the surface ellipsoid closest to the robot ($\mathcal{E}_m^*$) from Eq. (7.6) and evaluate Eq. (7.9) between the robot ellipsoid $\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P})$ and $\mathcal{E}_m^*$. However, this approach yields a non-smooth collision probability field due to a rapid spatial change in $\mathcal{E}_m^*$. Such a probability field may not be useful for optimization-based motion planning under uncertainty [212].

To mitigate this problem, we propose blending the collision probabilities of the K nearest surface ellipsoids from the robot ellipsoid. Such neighbors can be efficiently queried from spatial partitioning data structures such as KD-Tree [90]. In this work, we empirically set $K = 3$ for 2D and $K = 9$ for 3D workspaces. The final probability is given by

$$P^*(\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P}), f_Z) = \frac{1}{\sum_k w_k} \sum_k w_k P(\mathcal{E}_\mathcal{P}(\mathbf{p}, \mathbf{P}), \mathcal{E}_k^*), \tag{7.10}$$

where w_k denotes the blending weight of the k-th ellipsoid $\mathcal{E}_k^*$ in the set of nearest K ellipsoids $\{\mathcal{E}_1^*, \ldots, \mathcal{E}_K^*\}$. The blending weight w_k is the dot product of ∇d_k^* and the eigenvector directed towards the robot position, $\hat{\mathbf{n}}_k^*$, corresponding to the minimum eigenvalue of the shape matrix (Fig. 7.1). The dot product will be negative in cases when the angle between $\hat{\mathbf{n}}_k^*$ and ∇d_k^* is in the range $[\pi/2, 3\pi/2]$. For example, this situation can arise when ellipsoids represent a sharp turn. In such cases, we ignore the contribution of the ellipsoid to the collision probability (i.e., $w_k = 0$).

7.2 Results

This section presents an evaluation of the computational cost and accuracy of the proposed methods.

Targeting single-threaded operation on a CPU, the proposed methods are implemented in C++ using the Eigen library [78] with the highest level of optimization enabled in the GNU GCC compiler (`-O3`). The performance of this implementation is measured on several desktop and embedded platform CPUs to reflect the applicability of the proposed methods for a wide range of robots (Sect. 7.2.1).

The accuracy of the proposed continuous distance field is compared with the state-of-the-art continuous GP-based approach by Le Gentil et al. [107]. The accuracy measures for both the distance and gradient prediction are detailed in Sect. 7.2.2. For the proposed collision probability calculation method, the improvement due to the blending approach (Eq. (7.10)) is analyzed for different levels of noise in robot position. Experiments are conducted using 2D (Sect. 7.2.3) and 3D (Sect. 7.2.4) point clouds. For 3D, both simulated and real-world point clouds are used for evaluation.

7.2.1 Computational Performance

For this analysis, 100000 pairs of random ellipsoids are created by randomizing axis lengths, positions, and rotations of 3D ellipsoids. Axis lengths and positions (in all

directions) are sampled uniformly from the intervals $[0.1, 0.5]$m and $[-10, 10]$m, respectively. Random rotation matrices are generated using the Haar distribution [170]. Time taken for ellipsoid pair data structure initialization, distance and gradient computation, and collision probability calculation is measured. The initialization procedure caches matrices $\hat{\mathbf{B}}, \mathbf{\Lambda}_B, \mathbf{B}^{1/2}, \mathbf{B}^{-1/2}$, and $\mathbf{B}^{-1}$ for $\mathcal{E}_1(\mathbf{b}, \mathbf{B})$ (and the respective counterparts for $\mathcal{E}_2(\mathbf{c}, \mathbf{C})$), as defined in Proposition 1. Distance and gradient computation times correspond to calculating Eq. (7.4). For collision probability calculation, Eq. (7.9) is used. The performance is measured on one desktop (Intel i9-10900K) and three embedded platforms (NVIDIA Orin AGX, Orin NX, and Orin Nano).

Mean and standard deviation statistics of elapsed time (in microseconds) are reported in Table 7.1. It can be concluded that the time taken by all computers in Table 7.1 is between 10 to 60 microseconds. The total time (last column) represents the worst case where new ellipsoids are initialized every time the distance, gradient, and collision probability are calculated. In practice, the ellipsoids are initialized and updated using a few sensor observations; not for every distance or collision probability computation. As expected, the computation time increases as the clock speed of the CPU decreases.

Since real-world motion planning systems largely run on CPUs, these observations indicate that the proposed methods may be beneficial for adapting many existing motion planners towards leveraging GSMs as the environment representation. Furthermore, since the proposed methods run efficiently on a CPU, the GPU can be used for tasks like online high-fidelity RGB-D reconstruction. Note that fast fitting of GSMs to point cloud data is an active area of research, with recent approaches demonstrating high frame-rate operation on embedded computers like the NVIDIA TX2 [111].

	Time Taken (10^{-6}s), single-threaded execution			
Device (CPU)	Init.	Dist. + Grad.	Coll. Prob.	Total
i9 3.7 GHz	9.0 ± 1.7	9.5 ± 2.0	2.2 ± 0.9	20.7 ± 3.0
AGX 2.2 GHz	18.4 ± 1.1	17.3 ± 1.8	5.3 ± 0.6	41.0 ± 2.3
NX 2.0 GHz	22.4 ± 4.2	22.9 ± 2.6	6.7 ± 0.8	52.0 ± 5.2
Nano 1.5 GHz	24.5 ± 1.6	24.8 ± 2.7	7.0 ± 0.8	56.2 ± 3.4

Table 7.1: Mean and standard deviation of the time taken (in microseconds) to initialize a pair of ellipsoids and estimate distance, gradient, and collision probability using single-threaded execution on various CPUs.

7.2.2 Accuracy Measures

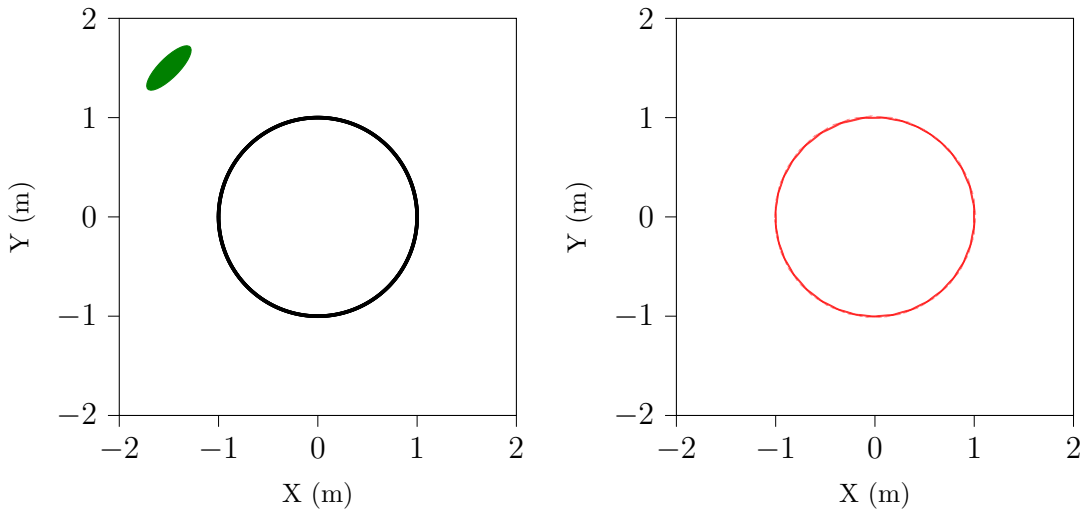
The accuracy in the distance field is measured (in meters) using the Root-Mean-Squared-Error (RMSE) between the prediction and the ground truth. A lower RMSE

indicates higher accuracy. For the gradient of the distance field, we use

$$1.0 - \text{RMSE}(\cos(\mathbf{D}_p, \mathbf{D}_g))$$

, where the cosine is computed at each corresponding point in the predicted ($\mathbf{D}_p$) and the ground truth vector field ($\mathbf{D}_g$) [140, Eq. 15]. This score is referred to as the Cosine Error Score (CES) in the following sections. A lower CES implies a better alignment between the predicted and the ground truth gradient.

The ground truth distance between ellipsoid and point cloud is computed using the point cloud distance between points densely sampled on the robot ellipsoid surface and the surface point cloud. For 2D numerical experiments, the ground truth gradient is calculated using the finite difference method on the ground truth distance field. For 3D experiments, the ground truth gradient vector is given by the surface normal at the point in the surface point cloud that is closest to the robot point cloud.



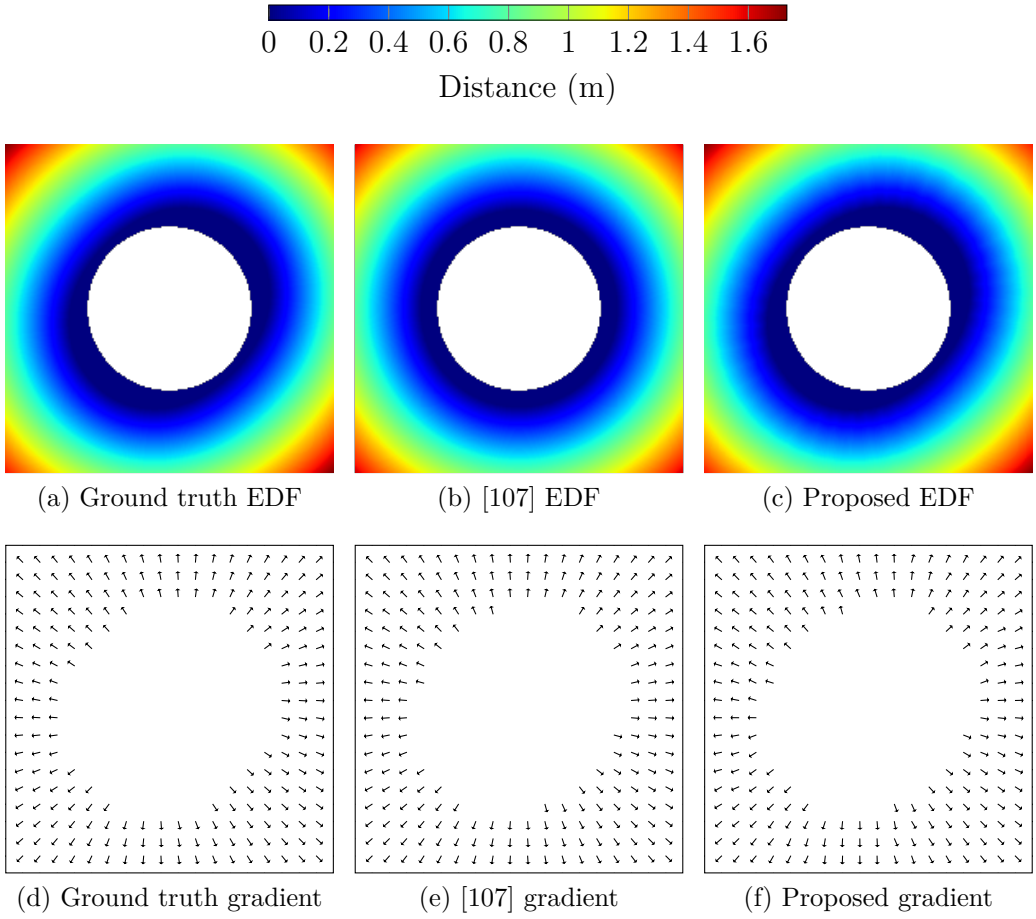
(a) Point cloud and robot

(b) GMM

Figure 7.2: Two-dimensional numerical scenario used in Sect. 7.2.3. (a) The input point cloud along with the representative position of the robot body model (green ellipse). (b) The GMM of the point cloud.

7.2.3 2D Experiments

The experimental setup is shown in Fig. 7.2. A point cloud with 1000 points is generated by uniformly sampling the boundary of a circular obstacle of radius 1.0 m. This point cloud represents a set of range measurements of the obstacle. The robot body is given by an ellipsoid with semi-axes lengths $(0.3, 0.1)$ meters and a rotation by 45 degrees from the positive x-axis (Fig. 7.2a). The GMM shown in Fig. 7.2b contains $M = 40$ components and it is generated from the point cloud using the approach from [76].



(a) Ground truth EDF (b) [107] EDF (c) Proposed EDF

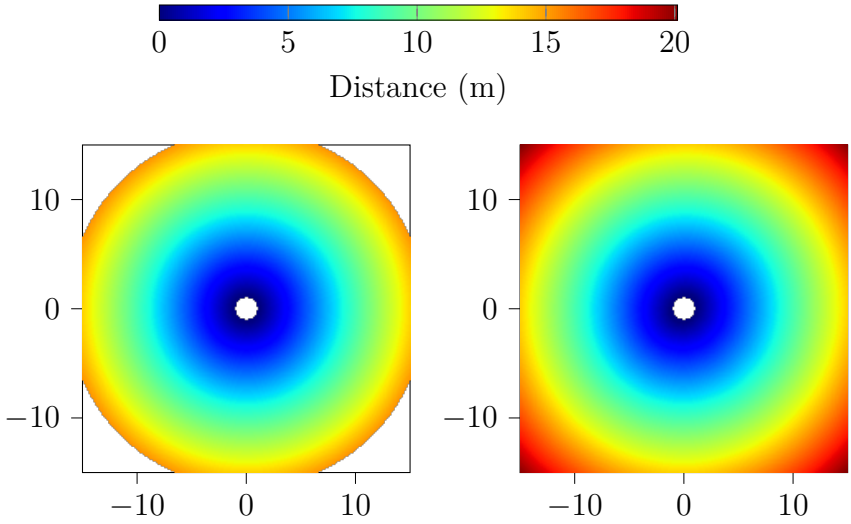
(d) Ground truth gradient (e) [107] gradient (f) Proposed gradient

Figure 7.3: Comparison of EDF and gradient accuracy for the 2D scenario in Fig. 7.2a. Qualitative difference for EDF can be observed in (a), (b), and (c) and for the gradient in (d), (e), and (f). *This figure is best viewed in color.*

Baseline [107]			Proposed			
l	RMSE (m) ↓	CES ↓	σ	M	RMSE (m) ↓	CES ↓
0.5	0.073	**0.003**	0.1	40	**0.007**	**0.003**
0.4	0.080	**0.003**	0.2	19	0.030	0.009
0.3	0.093	**0.003**	0.3	14	0.055	0.017
0.2	0.103	**0.003**	0.4	12	0.072	0.018
0.1	0.111	**0.003**	0.5	7	0.186	0.044

Table 7.2: Quantitative analysis for errors in EDF and gradient at different hyperparameter values. The best RMSE and CES values are bolded. The proposed method enables higher EDF accuracy compared to the baseline.

The number of components M may vary with the bandwidth hyperparameter σ in [76]; therefore, in quantitative results five different bandwidth values are considered. The baseline approach [107] uses the raw point cloud in all cases but requires a



(a) [107] Numerical Instability (b) Proposed EDF

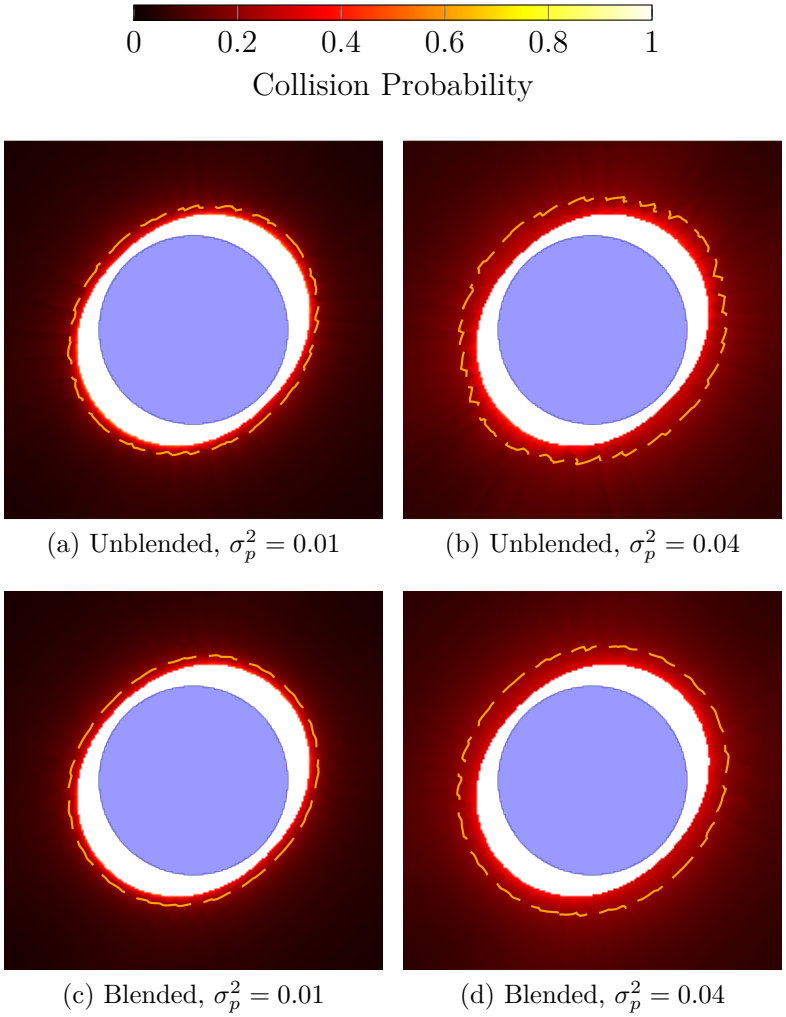
Figure 7.4: Heatmaps for EDF far from the surface generated using the baseline (a) and proposed (b) methods. The blank space at the edges in (a) shows the region where the EDF is undefined. Contrary to the baseline, the proposed approach is numerically stable in large workspaces.

hyperparameter called characteristic length l. It is stated in [107] that specifying this parameter is an open area of research. Therefore, the RMSE and CES values are computed for a set of σ and l parameters.

The center position of the robot body ellipsoid is varied on a 200×200 uniform grid in a workspace with extents $[-2.0, 2.0]$ meters in both x and y directions. At each robot position the Euclidean distance and its gradient from the surface point cloud are measured, resulting in EDFs and gradient vector fields in the 2D workspace for the ground truth, baseline, and proposed methods (Fig. 7.3).

Qualitatively, the distance isocontours appear rotated by 45 degrees in the ground truth EDF (Fig. 7.3a) because of the ellipsoid's fixed orientation. The baseline approach implicitly assumes a circular robot body of radius equal to the semi-major axis length of the ellipsoid (0.3 m). This results in a conservative EDF estimate (Fig. 7.3b) relative to the ground truth. However, the proposed approach (using the GMM in Fig. 7.2b) accounts for the ellipsoid robot model explicitly, resulting in a relatively accurate EDF (Fig. 7.3c). Notably, this EDF accuracy improvement is achieved while maintaining a similar level of accuracy in the gradient of EDF (see the arrows in Figs. 7.3d–7.3f).

The quantitative results are summarized in Table 7.2. In the baseline case, for decreasing l the EDF accuracy increases but the gradient accuracy is observed to be constant. For the proposed method, the EDF and gradient accuracy increases with decreasing σ. The best performing case for the proposed method enables about $10\times$ more accurate EDF prediction than the best performing baseline while achieving the same gradient accuracy. If a higher σ is used, the EDF estimates remain conservative



(a) Unblended, $\sigma_p^2 = 0.01$ (b) Unblended, $\sigma_p^2 = 0.04$

(c) Blended, $\sigma_p^2 = 0.01$ (d) Blended, $\sigma_p^2 = 0.04$

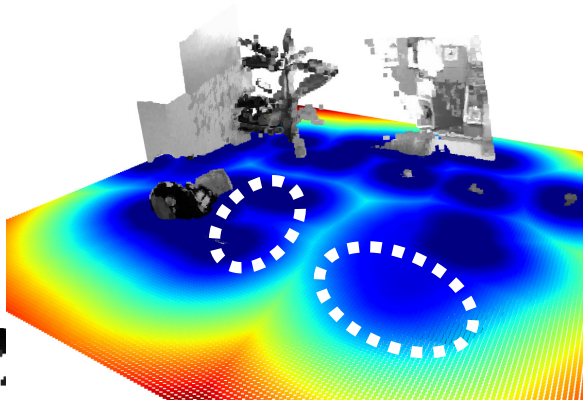
Figure 7.5: Heatmaps for collision probabilities computed using the proposed approach at different noise levels. The blue circle is the obstacle surface from Fig. 7.2a. From (a) to (b) and from (c) to (d) the noise increases and so do the collision probabilities (larger red regions in (b) and (d)). The dashed orange lines are isocontours at 15% collision probability. In the blended case the isocontour lines are smoother compared to the unblended approach. *This figure is best viewed in color.*

so the robot still remains safe.

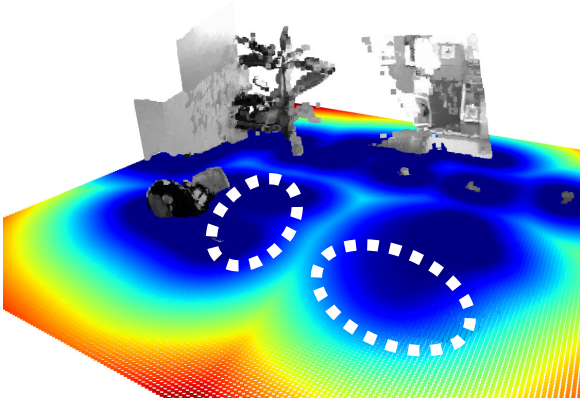
Figure 7.4 provides a comparison of the numerical stability during EDF computation far from the surface. The baseline approach requires a logarithm computation of an occupancy value that gets close to zero as the distance from the surface increases. Therefore, at a certain distance the estimates are invalid as $\log(0) \to -\infty$. In Fig. 7.4a, this effect can be seen in the blank areas nearly 10 m away from the surface where the EDF is not defined. In contrast, the proposed approach can estimate distance everywhere in the workspace as it depends on the distance between ellipsoids (Fig. 7.4b).

Let the center of the robot ellipsoid be Gaussian-distributed with the spherical covariance $\sigma_p^2 \mathbf{I}_2$, where $\mathbf{I}_n$ denotes the identity matrix of order n. The unblended

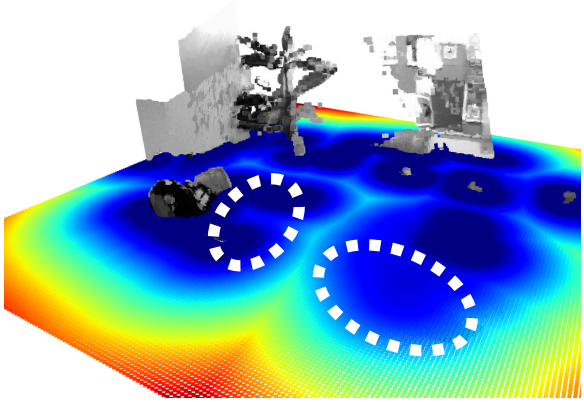
(a) Point Cloud



(b) Ground Truth



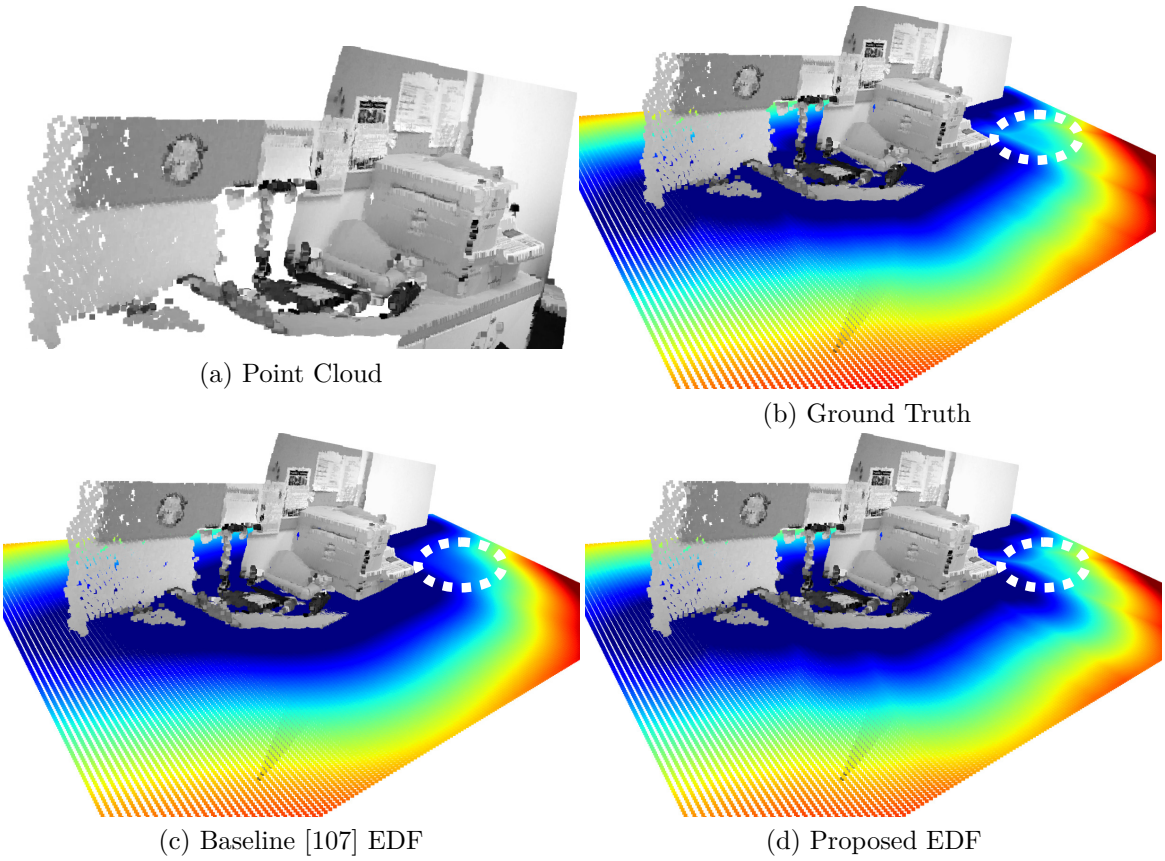
(c) Baseline [107] EDF



(d) Proposed EDF

Figure 7.6: Heatmaps for (b) ground truth, (c) baseline [107], and (d) proposed EDFs generated using the real-world 3D point cloud shown in (a). Note the difference in baseline and proposed EDFs relative to the ground truth in the dashed white regions. The dark blue regions are bigger in the baseline demonstrating conservative EDF estimation due to an implicit spherical robot body assumption. The proposed approach accounts for the ellipsoidal robot body while enabling continuous-space queries. *This figure is best viewed in color.*

and blended collision probabilities for $\sigma_p^2 = 0.01$ and 0.04 along with isocontours for probability level 0.15 are shown in Fig. 7.5. It is observed that for increasing levels of noise there is an overall increase in collision probabilities and the blending approach yields a smoother collision probability field. Both observations imply that the blended collision probability calculation method may be used for continuous-space queries in uncertainty-aware motion planning frameworks [212, 115].



(a) Point Cloud

(b) Ground Truth

(c) Baseline [107] EDF

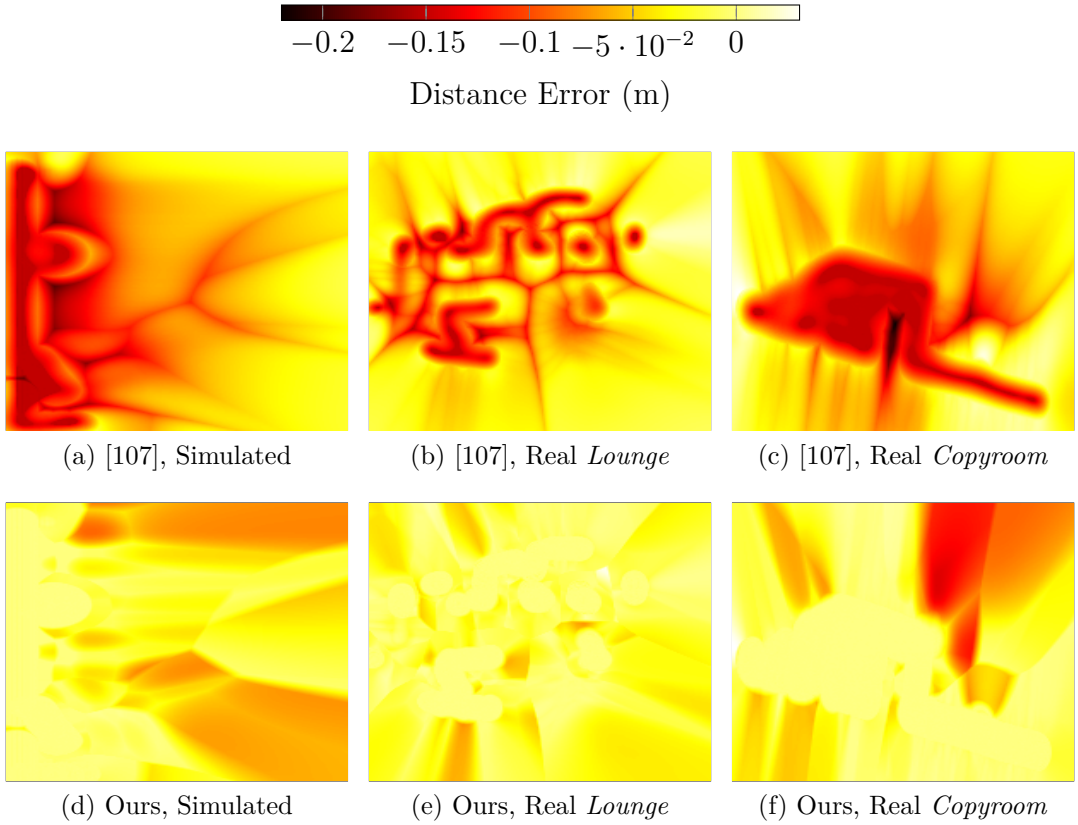
(d) Proposed EDF

Figure 7.7: Heatmaps for (b) ground truth, (c) baseline [107], and (d) proposed EDFs generated using the real-world 3D point cloud shown in (a). Note the difference in baseline and proposed EDFs relative to the ground truth in the dashed white regions. The dark blue regions are bigger in the baseline demonstrating conservative EDF estimation due to an implicit spherical robot body assumption. The proposed approach accounts for the ellipsoidal robot body while enabling continuous-space queries. *This figure is best viewed in color.*

7.2.4 3D Experiments

Three point clouds are used in 3D experiments: a simulated point cloud from the Living Room dataset [36], a real-world point cloud from the Lounge dataset [210], and a real-world point cloud from the Copyroom dataset [210]. The point clouds are constructed using two 320×240 RGB-D frames from each dataset (see Figs. 1.3a, 7.6a and 7.7a). The accuracy of EDF and its gradient is studied for 2D uniform grid (200×200) slices of these environments, as done in prior work [196, 107]. The robot is a 3D ellipsoid with semi-axis lengths $(0.15, 0.15, 0.07)$ and it is rotated about the z-axis by 45 degrees. Figures 1.3c and 1.3d show EDF and collision probability outputs for the proposed approach on the simulated point cloud.

Figures 7.6 and 7.7 show qualitative comparison of the EDF obtained from different methods for the real point clouds. We observe the same outcome as in the 2D experiments; the proposed approach is relatively accurate because it accounts for



(a) [107], Simulated (b) [107], Real *Lounge* (c) [107], Real *Copyroom*

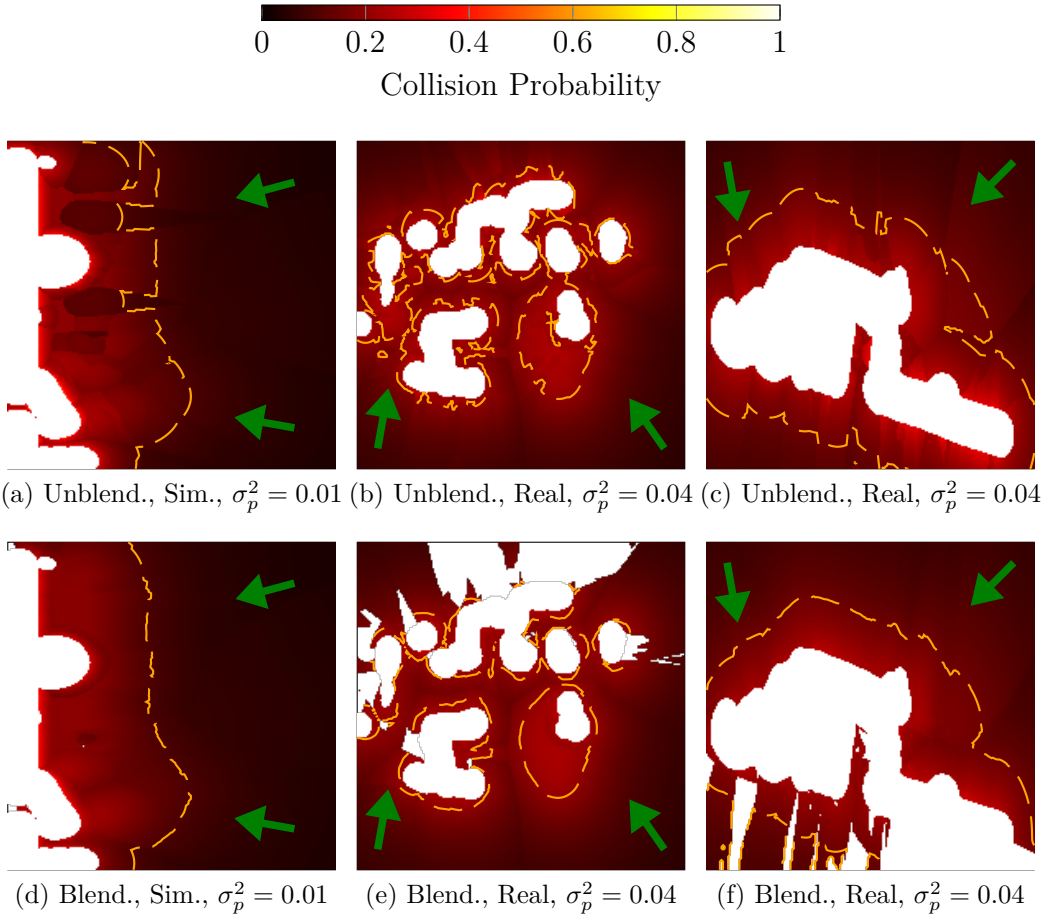
(d) Ours, Simulated (e) Ours, Real *Lounge* (f) Ours, Real *Copyroom*

Figure 7.8: Distance field estimation error heatmaps for the baseline and proposed approaches. The proposed approach enables a lower estimation error compared to the baseline.

the ellipsoidal robot body explicitly. For a better visualization of the difference in errors incurred by the two methods, Fig. 7.8 contains error heatmaps for the EDFs in Figs. 7.6 and 7.7.

For different values of hyperparameters the quantitative comparison of EDFs and gradient is summarized in Table 7.3. The range of hyperparameter l for the baseline approach is chosen by grid search to avoid numerical instability (Fig. 7.4). For σ, the range of values are based on the results in [76]. It is observed that for both simulated and real-world cases, the proposed approach enables relatively accurate EDFs at all hyperparameter levels while providing a comparable gradient accuracy.

Lastly, for collision probabilities, the experiment from the 2D evaluation (Fig. 7.5) is conducted for the given 3D point clouds (Fig. 7.9). A spherical covariance of $\sigma_p^2\mathbf{I}_3$ is used for the robot position uncertainty. As expected, for both simulated and real-world point clouds the collision probabilities decrease as the distance from the surfaces increases. Moreover, it is observed that in the occluded regions (i.e., regions behind the surface when viewed along the green arrows in Fig. 7.9) the blended approach does not provide reliable estimates. This is because the dot product between the surface normals and distance vectors is negative. It is reasonable to expect the estimates in the occluded regions to be degraded until the occluded regions are observed. The



(a) Unblend., Sim., $\sigma_p^2 = 0.01$ (b) Unblend., Real, $\sigma_p^2 = 0.04$ (c) Unblend., Real, $\sigma_p^2 = 0.04$

(d) Blend., Sim., $\sigma_p^2 = 0.01$ (e) Blend., Real, $\sigma_p^2 = 0.04$ (f) Blend., Real, $\sigma_p^2 = 0.04$

Figure 7.9: Unblended and blended collision probability fields over 2D slices of 3D simulated and real-world point clouds. More noise in robot position is added to the real-world cases to simulated the effect of higher position uncertainty during real-world deployments. Dashed lines show 10% probability isocontours. Green arrows show the directions of camera frustums from which the surface point cloud data is collected. The blending approach produces smoother collision probability estimates while ignoring occluded regions. *This figure is best viewed in color.*

unblended approach provides estimates without this consideration which may be risky during navigation. The blending approach results in smoother probability isocontour lines (lower noise in dashed orange lines) in the visible of the 3D experiments, also, making the approach valuable for continuous-space queries in future 3D uncertainty-aware motion planning frameworks.

7.3 Summary

This chapter detailed collision probability, Euclidean distance and gradient estimation for an ellipsoidal robot from a surface that is represented as a set of ellipsoids derived from Gaussian distributions. Prior work in ellipsoid-to-ellipsoid distance estimation

	Simulated		*Lounge*		*Copyroom*	
Approach	RMSE ↓	CES ↓	RMSE ↓	CES ↓	RMSE ↓	CES ↓
[107] (0.1)	0.054	0.166	0.039	0.258	0.056	**0.277**
[107] (0.2)	0.084	0.168	0.061	0.258	0.075	0.278
[107] (0.3)	0.119	0.173	0.098	0.258	0.100	0.283
[107] (0.4)	0.145	0.171	0.132	**0.257**	0.129	0.290
Ours (0.02)	0.042	**0.159**	**0.023**	0.263	**0.046**	0.289
Ours (0.03)	**0.038**	0.164	0.042	0.263	0.058	0.295
Ours (0.04)	0.068	0.181	0.038	0.267	0.057	0.305
Ours (0.05)	0.085	0.191	0.054	0.267	0.062	0.296

Table 7.3: Quantitative results using simulated and real 3D point clouds at different hyperparameter settings for the baseline and proposed methods. The best RMSE and CES values for each dataset are bolded.

was extended to compute distance and gradient in the proposed context. A geometrical blending approach ensured that the estimated collision probabilities are smooth so that they can be used for uncertainty-aware motion planning. These methods were validated using 2D and 3D real-world point cloud environments, demonstrating superior performance (as much as 10× in the 2D case) compared to the state-of-the-art continuous space method.

There are two key limitations of this work. First, the computation in Eq. (7.6) may require additional local submap extraction or spatial partitioning data structures to enable scalability as M increases. A concurrent or vectorized implementation of the eigenvalue problems in Proposition 1 may further improve performance. Second, estimation in the orientation space (formally, in the special orthogonal group $\mathbb{SO}(2)$ or $\mathbb{SO}(3)$) is not explicitly considered in this work which may be an interesting direction for future research.

These algorithms and methods address the challenge **C4** towards the thesis objectives.

CHAPTER 8
Active Multi-Robot Reconstruction using GMMs

This chapter details results from real-world experiments through single- and multi-robot field deployments equipped with motion primitives-based planning and GMM-based mapping. These evaluations are carried out to test the planning and mapping approaches from previous chapters with respect to communication efficiency.

For the single-robot deployment (Sect. 8.1), the motion primitives-based planner from Chapter 4 is employed with MCTS being restricted to only one level (i.e., a single-stage approach) to further reduce the computational requirements. The GMM-based point cloud modeling technique from prior work (Sect. 3.2, [180]) is used for mapping. The reconstruction performance, reconstruction accuracy, and communication-efficiency are compared with an occupancy grid mapping approach through real-world experiments in two caves.

For the multi-robot deployment (Sect. 8.2), the motion primitives-based planner from the single-robot deployment is extended to the multi-robot case through an inter-robot collision avoidance strategy. A distributed GMM-based mapping methodology is developed leveraging the insights in communication-efficiency from the single-robot deployment. The maximum speeds attained by the robots is characterized and the communication-efficiency of the two-robot team is compared with discretized approaches through experimentation in a wild cave in West Virginia, USA.

For all the deployments in this chapter a monocular visual-inertial navigation system is used to provide state estimates for precise control. Multirotor aerial platforms are utilized for all experiments. A summary of this chapter is provided in Sect. 8.3.

8.1 Single-Robot Deployment

The single-robot system (Fig. 8.1) is deployed in total darkness at Laurel Caverns[1] (Sect. 8.1.1), a commercially operated cave system in Southwestern Pennsylvania con-

[1] http://laurelcaverns.com/

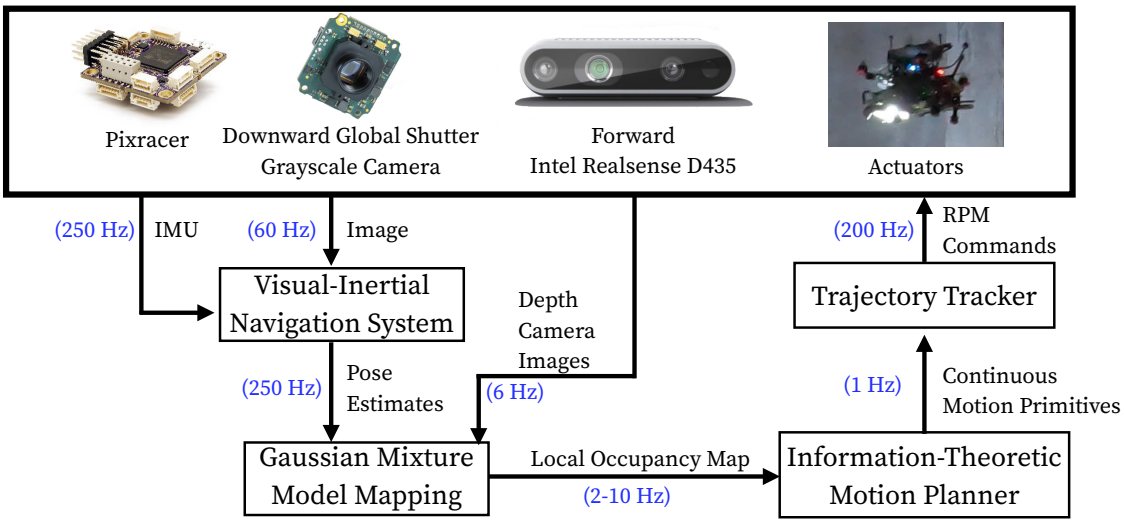


Figure 8.1: Overview of the autonomous reconstruction system presented in this chapter. Using pose estimates from a visual-inertial navigation system and depth camera observations, the mapping method builds a memory-efficient approximate continuous belief representation of the environment while creating local occupancy grid maps in real-time. A motion primitives-based information-theoretic planner uses this local occupancy map to generate snap-continuous forward-arc motion primitive trajectories that maximize the information gain over time.

sisting of over four miles of passages[24] and a wild cave in West Virginia[5] (Sect. 8.1.2).

8.1.1 Laurel Caverns

Figure 8.2a illustrates a composite image from several still images of the robot exploring the Laurel Caverns Dining Room. Two experiments were conducted, one

[2]The authors acknowledge that caves are fragile environments formed over the course of tens of thousands to millions of years. Laurel Caverns was chosen as a test site because it has relatively few speleothems[3]due to its sandstone overburden and the high silica content of the Loyalhanna limestone [143]. The authors worked with cave management to select a test site that contained low speleothem growth to minimize risk of damage to the cave. Cave management monitored all flights. No flights were executed near delicate formations.

[3]Speleothems are mineral formations found in limestone caves (e.g., stalagmites, stalagtites, and flowstone) that are composed of calcium carbonate, precipitated from groundwater that has percolated through adjacent carbonate host rock [20].

[4]Bat populations in the northeastern U.S. have been decimated with the onset of White-nose Syndrome in the winter of 2007-2008 [67]. Great care was taken not to disturb bats with the aerial systems during the hibernating season.

[5]The region of the cave where flight experiments were conducted contained speleothems that have ceased growing. Speleothems growth may terminate due to geologic, hydrologic, chemical, or climatic factors that cause water percolation to cease at a particular drip site [20]. The authors worked with cave management to select a test site that had neither actively growing speleothems or bats.

(a)

(b)

Figure 8.2: (a) A single aerial system explores the Dining Room of Laurel Caverns in Southwestern Pennsylvania. Still images of the robot exploring the environment are superimposed to produce this figure. (b) The aerial system with dimensions 0.25 m × 0.41 m × 0.37 m including propellers carries a forward-facing Intel Realsense D435 for mapping and downward-facing global shutter MV Bluefox2 camera (not shown). The pearl reflective markers are used for testing in a motion capture arena but are not used during field operations to obtain hardware results. Instead, a tightly-coupled visual-inertial odometry framework is used to estimate state during testing at Laurel Caverns.

for each of the MCG and OG approaches for a 95 s duration. The map entropy reduction over time is shown in Fig. 8.3c and is similar for both approaches, while the cumulative data transferred (Fig. 8.3d) to represent the maps is more than an order of magnitude lower for the MCG approach as compared to the OG approach. Note, however, that the communication reported for this experiment represents the theoretical, or estimated, communications needed to transmit the data. The data was not transmitted to a base station. The data transfer rate in Fig. 8.3e is calculated using Euler differentiation but note that the accuracy is affected by the limited number of samples. During hardware trials, a bounding box was used to constrain the reconstruction volume. To put the localization accuracy into perspective, the drift in position is about 0.53 m during a 50.9 m cave flight and the rotation drift is about 0.32 rad over 33.5 rad which is about a 1% drift in both translation and rotation. Position drift may be approximated as the difference between the initial and final

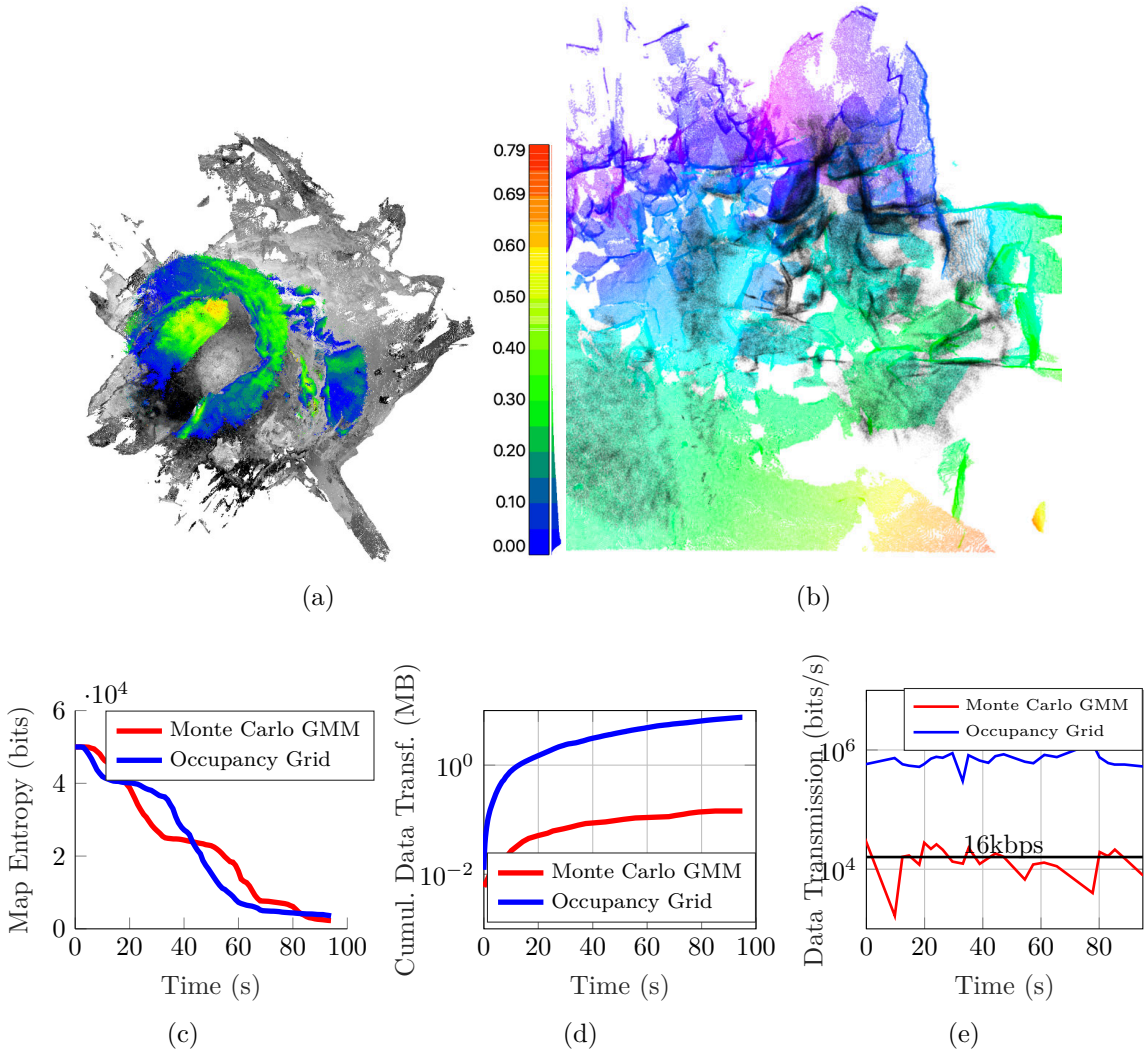


Figure 8.3: reconstruction statistics from the experiments at Laurel Caverns. (a) illustrates the reconstruction error of the resampled GMM map as compared to the FARO map by calculating point-to-point distances. The distribution of distances is shown on the right-hand side. The mean error is 0.14 m with a standard deviation of 0.11 m. In particular, there is misalignment in the roof due to pose estimation drift. (b) A subset of the resampled GMM map (shown in black) is overlaid onto the FARO map (shown in colors ranging from red to purple) that displays the breakdown in the middle of the Dining Room. (c) The entropy reduction and (d) cumulative data transferred for one trial for each of the Monte Carlo GMM mapping and OG mapping approaches are shown. The communication is a theoretical calculation – not actual transmitted data. While the map entropy reduction for each approach is approximately similar, the GMM mapping approach transmits significantly less memory than the OG mapping approach (0.1 MB as compared to 7.5 MB). (e) illustrates the bit rate for each approach in a semi-logarithmic plot where the vertical axis is logarithmic. The black line illustrates how the approaches compare to 16kbps. For comparison, 16kbps is sufficient to transmit a low resolution (176 × 144 at 5 fps compressed to 3200 bit/frame) *talking heads* video [40, 120].

position estimates because the robot takes off and lands at the same location.

8.1.2 West Virginia Cave

Figure 8.4a illustrates the map entropy reduction over time with a maximum duration of 150 s. The MCG and OG approaches perform similarly in these trials. The actual data transferred between the robot and base station is shown in Fig. 8.4b. Figure 8.4c is a composite image from several still images of the robot exploring the cave. A video of one reconstruction run may be found at the following link: https://youtu.be/H8MdtJ5VhyU. In these experiments a bounding box was used to constrain the reconstruction volume. The drift in position is about 1.28 m during the 82.6 m flight and rotation drift is about 0.55 rad over 41.8 rad which is about a 1.5% drift in position and 1.3% drift in rotation.

8.2 Multi-Robot Deployment

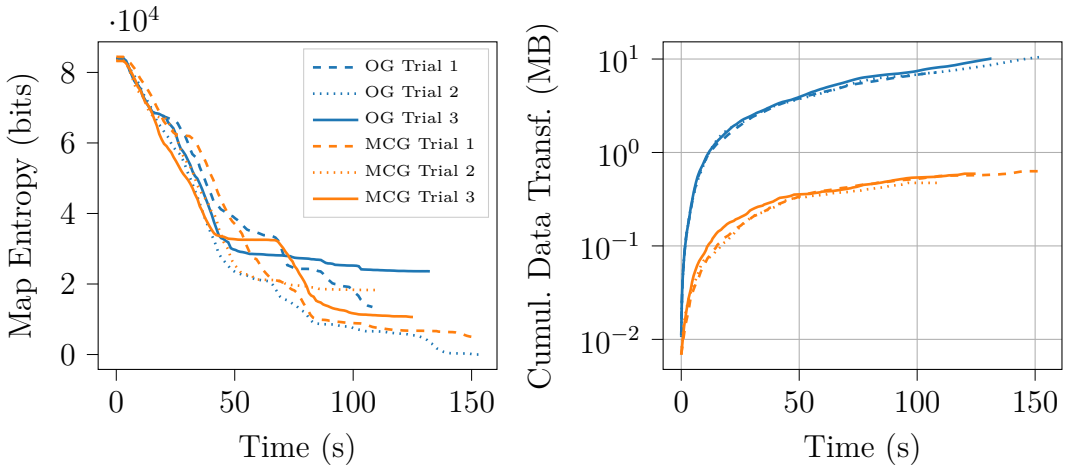
8.2.1 Approach

An overview of the system is shown in Fig. 8.5. Each robot is equipped with single-robot reconstruction and inter-robot communication modules. The reconstruction module consists of four major subsystems: GMM mapping, information-theoretic motion planning, visual-inertial state estimation, and trajectory tracking. The inter-robot communication module enables sharing information between robots or other computers on the network. The GMM mapping and planning subsystems together with the communication module constitute distributed mapping (Sect. 8.2.2) and multi-robot planning (Sect. 8.2.3), respectively. In this section, the following mathematical notation is used: lower-case letters represent scalar values, lower-case bold letters represent vectors, upper-case bold letters represent matrices, and script letters represent sets.

8.2.2 GMM-based Distributed Mapping

This section details the distributed mapping approach to share environment models between robots. Consider a team of N robots. At timestep t robot $i \in N$ receives the depth sensor observation, $\mathcal{Z}_t^i$, which represents a set of points. A Gaussian mixture model (GMM) is learned from these points following the approach from [180]. The GMM is parameterized by $\Theta = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^M$ where $\boldsymbol{\mu}_m \in \mathbb{R}^3$ is a mean, $\boldsymbol{\Sigma}_m \in \mathbb{R}^{3\times3}$ is a covariance, and $\pi_m \in \mathbb{R}$ is a weight such that $\sum_{m=1}^M \pi_m = 1$. A GMM representing point set $\mathcal{Z}_t^i$ is denoted as $\mathcal{G}$.

Keyframe GMMs. To reduce redundant observations, keyframe GMMs are identified for transmission to other robots. A keyframe GMM, $\hat{\Theta}_{\mathcal{Z}_t^i}$, is determined by approximating the field of view for the current sensor observation as a rectangular pyramid and calculating the overlapping volume with other keyframe fields of view. If the volume is smaller than a user-defined threshold, λ, the sensor observation is



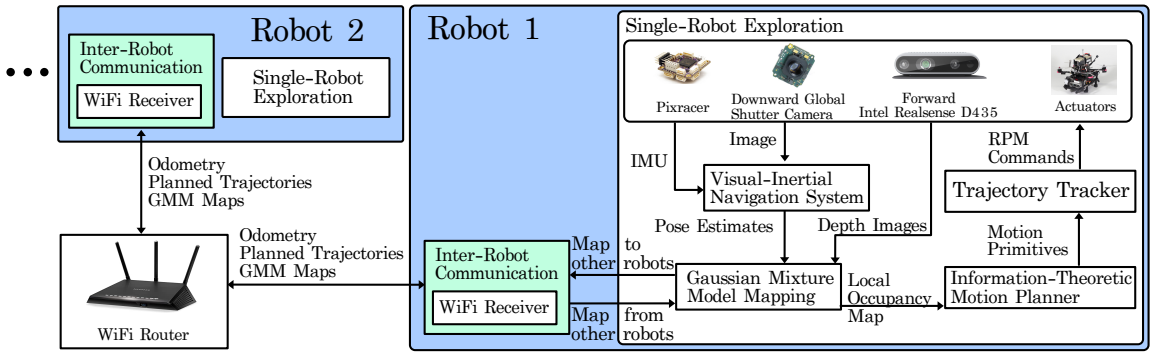
(a) (b)

(c)

Figure 8.4: Overview of the results from experiments in a cave in West Virginia. (a) The map entropy over time for three trials of the MCG and OG approaches. (b) The data transferred between a robot and base station for each trial. The communication reported is actual transmitted data over UDP to a base station. Note that while the reconstruction performance is similar for both approaches, the data transferred for the MCG approach is substantially less. (c) A composite image of one reconstruction trial composed of still images.

considered to be a keyframe. $\hat{\Theta}_{\mathcal{Z}_t^i}$ and the sensor pose, $\mathbf{S}_t^i \in \mathrm{SE}(3)$, are transmitted to the other robots or computers on the network.

Each robot maintains its own environment representation and relative initial transforms between robots are assumed to be known. When robot j receives $\hat{\Theta}_{\mathcal{Z}_t^i}$, it is received in the frame of robot i. To transform it into the frame of robot j, the relative initial rotation $\mathbf{R}_0^{ji} \in \mathbb{R}^{3\times3}$ and translation $\mathrm{T}_0^{ji} \in \mathbb{R}^3$ parameters are applied



(a) Multi-robot reconstruction framework

(b) Multirotors used in this work

Figure 8.5: (a) Overview of the rapid multi-robot reconstruction framework and (b) aerial systems used in experiments in this work.

(a) (b) (c)

Figure 8.6: Overview of the distributed mapping approach. (a) Robot i shown in red, takes a sensor observation shown in colors varying from red to purple and (b) learns a GMM (shown in red). If the GMM is determined to be a keyframe both the GMM and sensor pose are transmitted to robot j (shown in green). (c) The GMM and the sensor pose are transformed into the frame of robot j and used to update the occupancy.

to the means and covariances of the distribution using the following equations.

$$\boldsymbol{\mu}^j = \mathbf{R}_0^{ji}\boldsymbol{\mu}^i + \mathrm{T}_0^{ji} \qquad \boldsymbol{\Sigma}^j = \mathbf{R}_0^{ji}\boldsymbol{\Sigma}^i(\mathbf{R}_0^{ji})^T, \tag{8.1}$$

The transformed GMM is incorporated into robot j's existing GMM map following the approach from [180].

Occupancy Reconstruction. A local occupancy grid map $\mathbf{m}_t^i$ is maintained and centered around the robot's current position T_t^i for use in information-theoretic motion planning. To generate $\mathbf{m}_t^i$, a number of points $\mathbf{x} \in \mathbb{R}^3$ equal to the support

size, or number of points used to learn the distribution, is sampled and raytraced to the sensor pose T_t^i. The probability of occupancy along the ray is updated.

Multi-robot Map Updates. Care must be taken to update $\mathbf{m}_t^j$ when receiving $\hat{\Theta}_{\mathcal{Z}_t^i}$. In addition to applying the transformation parameters so that $\hat{\Theta}_{\mathcal{Z}_t^i}$ is transformed into the frame of robot j, $\mathbf{m}_t^j$ must also be updated by sampling points from the transformed $\hat{\Theta}_{\mathcal{Z}_t^i}$ and raytracing through $\mathbf{m}_t^j$ to the sensor pose, $\mathbf{S}_t^i$, which must also be transformed into the frame of robot j. This ensures the occupancy is updated with observations from both robots. A visualization of this is shown in Fig. 8.6. Robot i takes a sensor observation (Fig. 8.6a) and learns $\hat{\Theta}_{\mathcal{Z}_t^i}$ (Fig. 8.6b). This keyframe GMM is transmitted to robot j, transformed into the frame of robot j, and then used to update $\mathbf{m}_t^j$ (Fig. 8.6c).

8.2.3 Planning for Rapid Multi-Robot reconstruction

Robot i uses $\mathbf{m}_t^i$ for information-theoretic receding-horizon planning via the strategy presented in [73], which accounts for perception latencies and kinodynamic constraints of the robot. The approach uses Monte Carlo tree search (MCTS) [31] to evaluate the Cauchy-Schwarz Quadratic Mutual Information (CSQMI) [30] for a set of motion primitives over a user-specified time horizon. An informative primitive sequence is selected that maximizes the CSQMI over the MCTS tree. Safety is ensured by checking for collisions with the environment.

The informative trajectories are shared with other robots and inter-robot collision avoidance is enabled through a standard priority-based collision checker assuming a cylindrical robot model [24]. The priorities are assigned manually before the reconstruction run and remain constant throughout. To reduce the computational complexity for lower priority robots, three optimizations are applied. First, the collision checking is only active when a pair of robots are within a pre-specified radius. To enable this on each robot without assuming a centralized oracle, the robots share odometry information at a sufficiently high rate (10 Hz) compared to the planning frequency (1 Hz). Second, the number of cylinders sampled over the planned trajectory is limited to a pre-specified maximum to cap the number of cylinder-cylinder collision checks. This maximum value and the associated cylinder collision radius are selected conservatively based on the length of the motion primitive assuming the robot starts at hover and achieves a top speed at the endpoint. Third, for each robot the collision checks are performed only with the candidate motion primitive and the associated stopping motion primitive at the first depth of the MCTS tree because each depth of the tree is of a sufficiently long duration (2 s) as compared to the planning time (1 s).

8.2.4 Multi-Robot Experiments and Results

The experimental evaluation is motivated through a concept of operations for a multi-robot reconstruction mission in a Martian cave. Two robotic systems explore a Martian cave, transmit their maps to a surface station, which serves as a relay to an orbiter, and the orbiter transmits the data to operators on Earth. Three evaluations

are conducted to quantify the system performance through this concept of operations: first, the perceptual fidelity and memory usage of the map is compared to state-of-the-art approaches in a representative cave environment (Sect. 8.2.5); second, a hardware experiment is demonstrated with two rapidly exploring aerial systems and the communication requirement for each mapping approach is compared (Sect. 8.2.6); and third, a simulation study is conducted to study the effects of the bandwidth constraints on reconstruction performance (Sect. 8.2.7).

To correctly analyze the performance of the simulation study, the bottleneck in data transmission rate is identified and bounds on the rates are determined. In this scenario, data is transferred between the subterranean robot and surface station[6], surface station to orbiter[7], and orbiter to Earth[8]. The bottleneck in communication is between the subterranean robot and surface station when the robot is transmitting at depths between 20–25 m below ground, so the results in Sect. 8.2.7 are presented for the rates 0.1–0.25 Mbit/s, which are in line with data transmission rates at these depths. Throughout this section the shorthand OG is used to refer to the occupancy grid mapping approach [63] while OM refers to OctoMap [88].

8.2.5 Perceptual Detail Evaluation

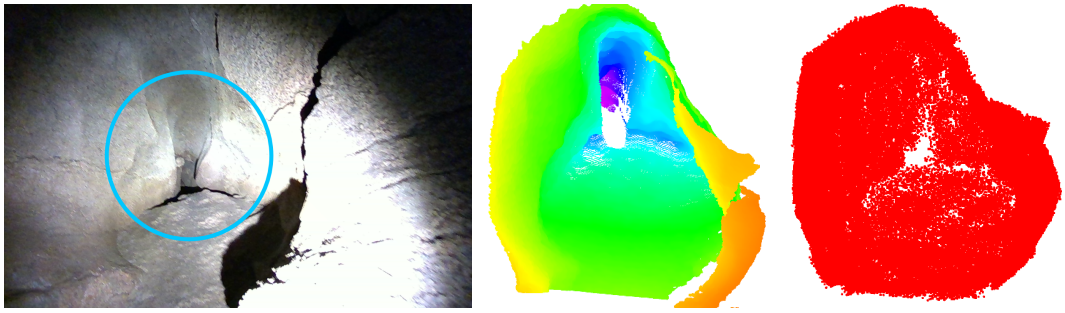
The first evaluation compares the perceptual fidelity of different environment representations in the context of memory usage. An RGB image and point cloud of a crevice in the cave are shown in Figs. 8.7a and 8.7b respectively. It is not clear from the image and depth information if the passage continues or there is a lack of data due to insufficient accuracy in the sensor observation. In either case, additional views are required to determine the exact nature of the passage. Figure 8.7g demonstrates that as the resolution of the OG and OM approaches increases, the memory demands also substantially increase. By comparison, the GMM approach requires substantially less memory. When using the GMM approach, the resulting resampled point cloud is shown in Fig. 8.7c, where a hole in the data is visible. This approach is compared to OM with varying leaf sizes in Figs. 8.7d–8.7f.

To obtain these results, a GMM was learned consisting of 100 components. Each component requires 10 floating point numbers which includes six floating point numbers to represent the symmetric covariance, three floating point numbers for the mean, and one floating point number to represent the mixing weight. Additional memory was used to represent the pose via six floating point numbers (three each for translation and rotation) where each floating point number is assumed to be four bytes. A 32-bit

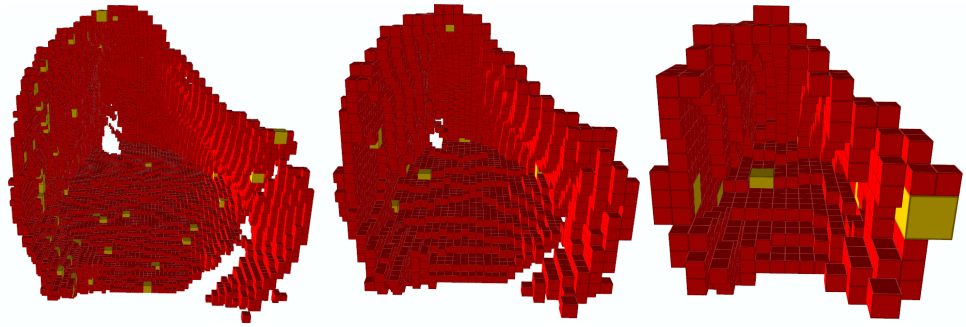
[6]Whittaker et al. [191] suggest the use of either very low frequency (VLF) radios or magneto-inductive (MI) links to achieve limited data rate through thick layers of rock. The MI links in particular can provide approximately 20-25 m dry soil penetration at channel capacity ranging from 0.1-0.25 Mbit/s when using small antennas (coils) [101]. In the results presented in Sect. 8.2.7, it is assumed that the robots could be equipped with these MI links.

[7]Orbiters can communicate at approximately 0.208-0.521 Mbit/s with a surface station for 8 minutes per sol, or Martian day [132].

[8]To transmit from the orbiter to Earth, the communication rate depends on which orbiter is above the lander to relay the data to Earth. The simulation study in Sect. 8.2.7 assumes the lowest data rate from the Mars Odyssey orbiter, which ranges from 0.128-0.256 Mbit/s [132].



(a) RGB Image (b) Point Cloud (c) Resampled GMM



(d) OM (0.025 m) (e) OM (0.05 m) (f) OM (0.1 m)

	0.025 m	0.05 m	0.1 m
(bytes)	(bytes)	(bytes)	(bytes)
GMM	**Occupancy Grid (OG)**		
4028	1.3×10^6	1.8×10^5	2.7×10^4
GMM	**OctoMap (OM)**		
4028	2.2×10^5	5.8×10^4	1.4×10^4

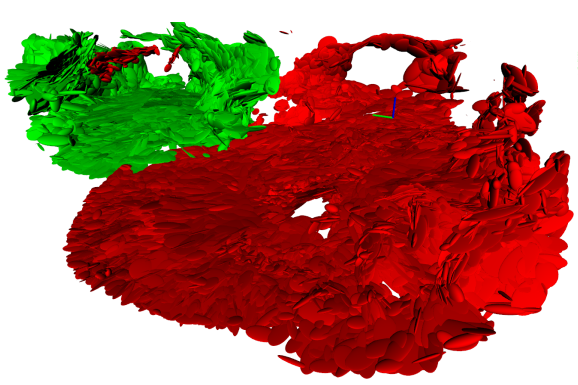
(g) Memory

Figure 8.7: Fidelity and memory usage evaluation of several mapping approaches. (a) and (b) illustrate data from a representative environment the robot may encounter in the cave. A potential passage is circled in cyan. (g) highlights significant reduction in memory usage required by the GMM approach as compared to the OG and OM approaches. (c) Resampled points from the GMM are shown in red. (d)–(f) illustrate the OctoMap representation with leaf sizes varying from 0.025 m to 0.1 m. Leaf voxels are shown in red and larger voxels in yellow.

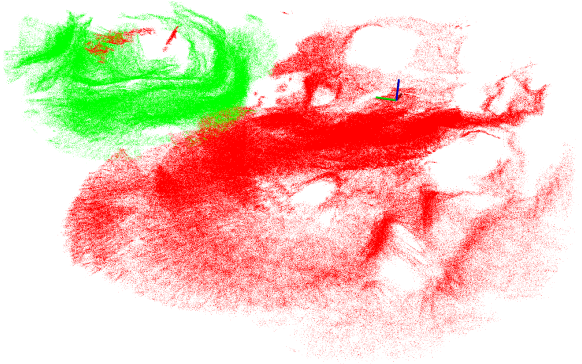
unsigned integer (four bytes) is also used to represent the support size of the GMM. In the OG case, one floating point number is used to store the logodds value and one unsigned integer (four bytes) is used to represent the index for each voxel in the change set. The total change set of N voxels is transmitted along with meta-data to reconstruct the grid. The meta-data consists of three unsigned integers to represent the dimensions of the grid in width, height, and length as well as three floating point numbers to represent the origin for a total of 24 bytes. The total data required to represent the sensor observation with an OG is $8N + 24$ bytes. For OM, the full

probabilistic model is serialized and stored to disk. The size of the file is reported in the table. The motivation for retaining the logodds values in the OG and OM representations is to enable information-theoretic planning.

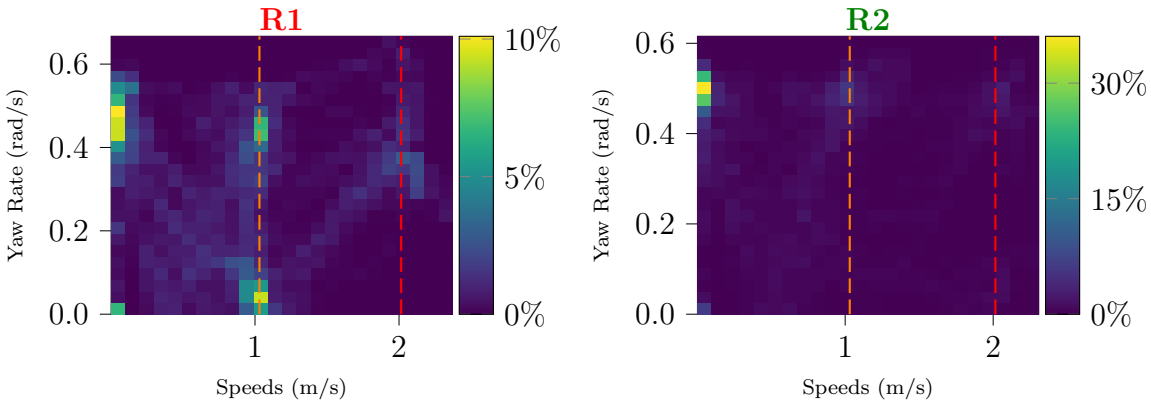
(a) Robots (circled) deployed in a cave. Communication router shown via dotted line.



(b) Combined GMM map



(c) Resampled points from the GMM map



(d) Speed bounds shown by dashed lines.

Figure 8.8: Rapid and communication efficient reconstruction of a cave with a team of two aerial robots. (a) illustrates the environment with the two robots (**R1** and **R2**) and the WiFi router used for communication. (b) illustrates the final GMM maps generated on the base-station. (d) shows the percentage density plots for linear speeds and yaw rates as measured by the visual-inertial navigation system during flight. A video of the flight can be accessed here: https://youtu.be/osko8EKKZUM.

The advantage of the GMM approach is that the probability of occupancy can be reconstructed at an arbitrary voxel resolution [179, 138], which significantly reduces the memory requirements as compared to the OG and OM approaches. The OG and OM approaches must retain the probability of occupancy to enable information-theoretic reconstruction [30, 207].

Table 8.1: This figure highlights that the GMM approach requires significantly less memory to represent the combined map as compared to state-of-the-art approaches. In the context of transmitting this data using a channel with capacity 0.25 Mbit/s, it would take significantly less time for the GMM approach as compared to the other approaches.

Completion %:	45%		65%		85%	
Case	Map Size (Mbit)	Time (hours)	Map Size (Mbit)	Time (hours)	Map Size (Mbit)	Time (hours)
GMM	0.7×10^{1}	8.0×10^{-3}	1.4×10^{1}	1.6×10^{-2}	1.9×10^{1}	2.2×10^{-2}
OG (0.1 m)	9.2×10^{1}	1.0×10^{-1}	1.57×10^{2}	1.7×10^{-1}	2.0×10^{2}	2.3×10^{-1}
OG (0.05 m)	5.4×10^{2}	6.0×10^{-1}	9.1×10^{2}	0.1×10^{1}	1.2×10^{3}	0.1×10^{1}
OG (0.025 m)	3.9×10^{3}	0.4×10^{1}	6.7×10^{3}	0.7×10^{1}	8.9×10^{3}	0.9×10^{1}
OM (0.1 m)	2.4×10^{2}	2.6×10^{-1}	3.9×10^{2}	4.4×10^{-1}	5.2×10^{2}	5.8×10^{-1}
OM (0.05 m)	1.6×10^{3}	0.2×10^{1}	2.6×10^{3}	0.3×10^{1}	3.4×10^{3}	0.4×10^{1}
OM (0.025 m)	9.8×10^{3}	1.1×10^{1}	1.6×10^{4}	1.8×10^{1}	2.1×10^{4}	2.4×10^{1}

8.2.6 West Virginia Cave

The second evaluation consists of hardware experiments for two aerial systems exploring the cave. The experiment demonstrates (1) each robot generates informative plans with linear speeds up to 2.37 m/s and yaw rates up to 0.6 rad/s while maintaining safety and (2) the communication required to transmit the map from robots to a base station is substantially less as compared to the OG and OM approaches. For the purposes of this experiment, the robots are deployed in disjoint bounding boxes and the coordination between robots is not studied. What follows is a description of the experimental setup (including the implementation details) and results.

Each robot in the multi-robot system employs the navigation and control technique outlined in prior work [180]. The robots communicate with other computers on the network via WiFi and use the User Datagram Protocol (UDP) to transfer packets over the network. Before the start of each experiment, the SE(3) transform between the takeoff positions of the robots is measured manually using the navigation approach. The relative initial transform is used by the distributed mapping subsystem to align the GMM map fragments in the frames of other robots to the current robot's local frame.

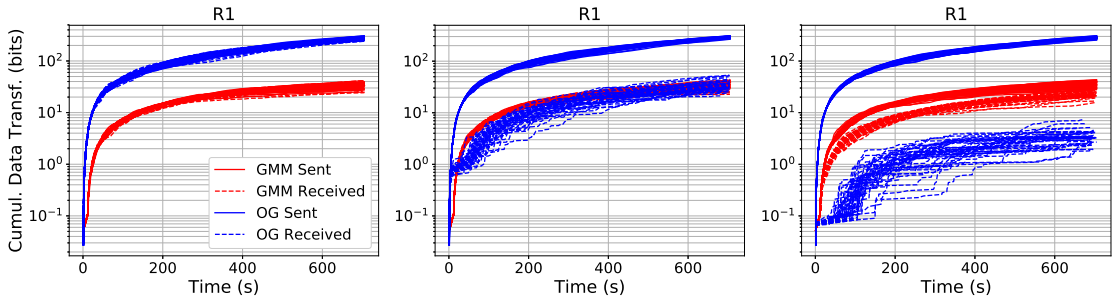
The maximum speed[9] of the robots in the xy-plane is 2.0 m/s, the maximum speed towards unknown space is 1.0 m/s, the maximum z-direction speed is 0.25 m/s, and the maximum yaw rate is constrained to 0.5 rad/s. One of the metrics used to assess the planning performance is quantifying the maximum speed and yaw rate achieved by the robot while ensuring collision free operation. Both linear and yawing motions are exploratory actions for an aerial robot equipped with a limited field of view depth

[9]The speed limits and the operational volumes were chosen based on the cave passage dimensions. The authors worked with cave management to select a test site that contained neither actively growing speleothems or bats. Possible effects of imperfect trajectory tracking and state estimation were also taken into account.

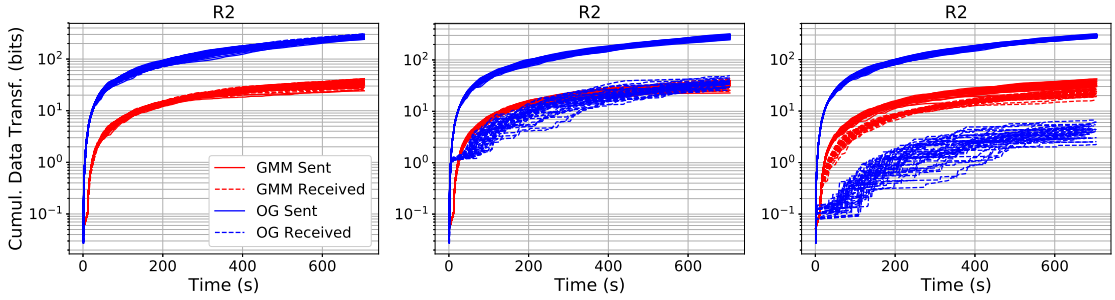
sensor [73, 180]. The data transmitted from the robots to the base station is used to quantify the success of the mapping approach. The GMM results of Fig. 8.8 are generated in flight during an actual trial in the cave. To enable a fair comparison, the depth images collected from the GMM reconstruction trial in the cave are post-processed using the OG and OM approaches. This ensures that variation in the other subsystems does not unduly affect the results. An analysis to quantify the memory required for each approach similar to Sect. 8.2.5 is presented. The OG and OM results are generated by updating the map using the depth information for the current image and publishing the change set. For the OM approach, the change set is serialized to file as the full probabilistic model to enable the base station and other robot to exactly recreate the map for information-theoretic reconstruction.

The two deployed robots are denoted by **R1** and **R2** in Fig. 8.8. The robots achieve high reconstruction rates by selecting actions that enable safe operation at linear speeds up to 2.37 m/s and yaw rates up to 0.6 rad/s, which are of the same order as state-of-the-art fast reconstruction works[10] [73, 49, 46]. Moreover, note that since **R1** operates in a relatively open space compared to **R2**, a larger percentage of high speed actions are selected (Fig. 8.8d). In contrast, the planner selects the yawing motion and slow linear actions towards frontiers more often for **R2** to allow for safe operation in a constrained space (Fig. 8.8d). Both of these behaviors in the multi-robot system arise automatically due to the choice of the action representation for single-robot planning in [73]. These behaviors show that the same action representation can be used on every robot in the team without any change in parameters and still allow for intelligent speed adaptation for rapid and safe reconstruction.

The combined map from **R1** and **R2** requires significantly less time to transmit under the bandwidth constraint when measuring at various points during reconstruction (Table 8.1). An implication of this in the context of the concept of operations is that at 100% reconstruction completion it will take about 104.40 seconds to transmit the GMM map, 12.30 hours to transmit the 0.025 m resolution OG map, and 1.25 days to transmit the 0.025 m resolution OM map to Earth. It is important to note why the OM approach requires more memory than the OG approach for this result while it required less memory than the OG approach in Fig. 8.7g. The change set must be encoded as an OctoMap before serializing to file. The approach presented by Hornung et al. [88] requires that the spatial relationships between nodes be implicitly stored in the encoding. This means that the serialized stream does not contain any 3D coordinates and additional data must be stored to preserve the structure of the octree. This is in contrast to the OG approach that stores a logodds value and index from which 3D coordinates can be recovered. Therefore, for small change sets, the OM approach has much higher overhead than the OG approach.



(a) R1 - No limit (b) R1 - 0.25 Mbit/s (c) R1 - 0.1 Mbit/s



(d) R2 - No limit (e) R2 - 0.25 Mbit/s (f) R2 - 0.1 Mbit/s

Completion %:	45%			65%			85%		
Comm. Limit	GMM (s)	OG (s)	Δ (%)	GMM (s)	OG (s)	Δ (%)	GMM (s)	OG (s)	Δ (%)
No limit	80.19	81.51	1.62	130.73	131.98	0.95	225.8	237.51	4.93
0.25 Mbit/s	79.91	92.38	**13.5**	129.1	160.15	**19.39**	214.86	282.11	**23.84**
0.1 Mbit/s	86.51	93.43	7.41	142.95	165.88	13.82	247.83	270.15	8.27

(g) reconstruction completion times

Figure 8.9: Variation of reconstruction performance with inter-robot communication limits. (a) to (f) plot the cumulative map data sent and received for the GMM and OG approaches under different data rate constraints for the two robots. The received data is impacted significantly for the OG approach at 0.25 Mbit/s while both approaches are affected at 0.1 Mbit/s. Note that in all experiments the planning and coordination methodology is kept the same for a fair comparison. (g) compares the time to achieve a certain percentage of environment coverage. We observe that at the 0.25 Mbit/s constraint, the GMM approach improves the performance of the team by up to 23.84%.

8.2.7 Effects of Constrained Communication

For this study the assumption on the robots operating in disjoint spaces is relaxed and a priority-based inter-robot collision checker is implemented for shared space operation. The simulation consists of a two-robot team that explores the cave environ-

[10]The attained speeds exceed the limits slightly due to imperfect trajectory tracking and state estimation.

ment. Two approaches are tested: GMM and OG. The OM approach is not compared for this experiment because to the best of our knowledge there is no existing open-source implementation of the Shannon mutual information used for planning by Zhang et al. [207]. Further, this enables us to retain the same planning subsystem for a fair comparison of the GMM and OG approaches. The communication rate is varied among 0.1 Mbit/s, 0.25 Mbit/s, and unconstrained. Each configuration is tested in 40 experiments with a 700 s duration. The duration of the reconstruction is chosen based on the top speed of the robots and the spatial dimensions of the environment. The reconstruction software is run on separate computers in a distributed fashion over a wired connection. The simulations are run on two desktop computers running Ubuntu 18.04 with Intel i7-6700K CPUs. One computer has 32 GB RAM and the other has 16 GB of RAM. For the wired connection, the data rate is limited via the network traffic control tool in Linux that uses the Token Bucket Filter (TBF) to maintain the specified rate value [89]. Figure 8.9 illustrates the results from the simulation study. As the communication bandwidth is reduced from no limit in Fig. 8.9a to 0.25 Mbit/s the OG approach begins to drop packets and the reconstruction performance of the multi-robot approach decreases as compared to the GMM approach (see Fig. 8.9g). At this rate, the GMM approach achieves 85% environment coverage in less than 80% of the time that it takes the OG approach. However, as the communication rate decreases further to 0.1 Mbit/s the GMM approach also suffers though it is able to outperform the OG approach.

8.3 Summary

This chapter presented field deployment results towards gains in communication-efficiency due to GMM-based mapping with fixed number of components. We demonstrate up to 100x improvements over discretized techniques like occupancy grid mapping and OctoMap while achieving a higher fidelity in reconstruction.

This chapter addresses the challenge **C5** towards the thesis objectives.

CHAPTER 9
Conclusion and Future Work

This thesis hypothesizes that:

> Using Gaussian mixture models as the map representation yields high-fidelity maps, communication-efficient map sharing, and safe informative planning via motion primitives during active reconstruction with multiple robots.

The key contributions of this thesis are summarized in Section 9.1 and several directions for future research are presented in Section 9.2.

9.1 Summary of Contributions

In Chapter 4, we created a motion primitives-based planner for active reconstruction for a quadcopter vehicle equipped with an RGB-D camera. This planner is anytime (i.e., it can pick suboptimal plans at any rate requested by the controller) and safe (i.e., the actions are always within the actuator constraints). The design of motion primitives library is informed by the common actions a robot fixed with a limited field-of-view sensor needs to take during reconstruction. The quadcopter was able to fly rapidly in the environment and capture a depth map in real-time.

In Chapter 5, we extended the work from Chapter 4 by making the maximum speed of the motion primitive library adapt with the resolution of the local map. The main benefit of this method is in navigating tight spaces in presence of varying amount of clutter. The robot automatically slows down and speeds up depending on the collision risk associated with the environment. Real world tests demonstrate that the proposed approach enables navigating through gaps in the environment that would otherwise not be visible if a fixed resolution planning approach was used.

In Chapter 6, the Self-Organizing Gaussian mixture modeling (SOGMM) approach was introduced. Instead of guessing the number of components based on prior heuristics (e.g., AIC, BIC), this work used a information-theoretic learning approach called the principle of relevant information to guess the number of components according to the complexity of range and intensity data. The resulting method provides higher-fidelity reconstruction at a much lower map size compared to Octomap [88] and Nvblox [128] approaches.

In Chapter 7, methods to estimate distance, distance gradient, and collision probability of an ellipsoidal robot from the SOGMM were presented. Compared to the dense occupancy grids or signed-distance field based methods, this method does not require propagating a wave function to compute a distance value at any point in space.

Instead, it uses an efficient approach for ellipsoid to ellipsoid distance estimation to trivially parallelize the distance calculation to an SOGMM without making a spherical robot assumption. The resulting approach provides benefits in both accuracy and memory efficiency compared to state of the art Gaussian Process-based methods.

Finally, in Chapter 8 we presented the distributed mapping and inter-robot collision avoidance methods that enable active reconstruction on low-bandwidth communication channels. Several field deployments in real-world caves demonstrate the efficacy of the approach.

9.2 Future Research

There are several directions for future work. We classify them into short-term and long-term goals below.

9.2.1 Short-Term Goals for Future Work

- Integration with prior works on registration [178] and loop closure [176] with GMMs to enable large-scale consistent mapping with SOGMMs.
- Instead of using an intensity image, use semantic features [100] in SOGMM. This would require testing the method with categorical distributions.
- Generate a scene graph using SOGMM as the underlying representation as opposed to a mesh [185].
- Pass the environment representation on telemetry radio as opposed to WiFi or any expensive/high-bandwidth connection. Currently there are 900 MHz radios that are used in multi-robot deployments that only provide low-bandwidth odometry or analog video. Given the level of compression that SOGMM provides, we should be able to transfer high-fidelity maps over 900 MHz radios.
- Extend the teleoperation methodology in Chapter 5 to multiple robots such that one human can teleoperate many robots while the robots pass the environment representation over long-range radios.
- Integrate visual-inertial odometry with SOGMM. Some initial work has been done in [70] for Gaussian splatting maps, but tight integration of an IMU is lacking.
- Extend the motion primitives-based planning method to articulated robots, leveraging prior work [204] and recent advances in diffusion model-based control [85].

9.2.2 Long-Term Goals for Future Work

- Replace volumetric mapping [88, 137, 156] with surface only maps using SOGMM in most robotics applications. This would require an extension to SOGMM towards more generally maintaining a boundary between known and unknown space.
- After integrating RGB into the SOGMM framework, propose a solution to the *kidnapped robot problem*. There has been an attempt for this on a smaller scale environment [51], but for arbitrary large environments with many robots, this problem remains a major challenge.
- SOGMMs can serve as feature inputs to transformers [188] via the cluster attention [124] mechanism. Several applications can then use SOGMM as the underlying feature input.
- Explore the theoretical connection between diffusion models and GMMs [79] to enable using diffusion within the SOGMM framework.
- Generalize SOGMM to include mixtures of Fisher distributions [206] to enable continuous inference of surface normals [84].
- Investigate theoretical properties of the mean shift algorithm [199, 200, 198] and see if the SOGMM method can be made certifiably robust to outliers [27].

This thesis has made progress on developing systems and methods communication-efficient active reconstruction using several robots in the real world. While the contributions indicate motion primitives-based planning and GMM-based mapping to be excellent tools, the future work directions above are equally exciting.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning., 2016. (see pages: 133, 141)

[2] Ali Agha, Kyohei Otsu, Benjamin Morrell, David Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, Amanda Bouman, Xianmei Lei, Jeffrey Edlund, Muhammad Ginting, Kamak Ebadi, Matthew Anderson, Torkom Pailevanian, Edward Terry, Michael Wolf, Andrea Tagliabue, Tiago Vaquero, Matteo Palieri, Scott Tepsuporn, Yun Chang, Arash Kalantari, Fernando Chavez, Brett Lopez, Nobuhiro Funabiki, Gregory Miles, Thomas Touma, Alessandro Buscicchio, Jesus Tordesillas, Nikhilesh Alatur, Jeremy Nash, William Walsh, Sunggoo Jung, Hanseob Lee, Christoforos Kanellakis, John Mayo, Scott Harper, Marcel Kaufmann, Anushri Dixit, Gustavo Correa, Carlyn Lee, Jay Gao, Gene Merewether, Jairo Maldonado-Contreras, Gautam Salhotra, Maira Saboia Da Silva, Benjamin Ramtoula, Seyed Fakoorian, Alexander Hatteland, Taeyeon Kim, Tara Bartlett, Alex Stephens, Leon Kim, Chuck Bergh, Eric Heiden, Thomas Lew, Abhishek Cauligi, Tristan Heywood, Andrew Kramer, Henry Leopold, Hov Melikyan, Hyungho Choi, Shreyansh Daftry, Olivier Toupet, Inhwan Wee, Abhishek Thakur, Micah Feras, Giovanni Beltrame, George Nikolakopoulos, David Shim, Luca Carlone, and Joel Burdick. NeBula: TEAM CoSTAR's Robotic Autonomy Solution that Won Phase II of DARPA Subterranean Challenge. *Field Robotics*, 2(1):1432–1506, March 2022. ISSN 27713989. doi: 10.55417/fr.2022047. (see page: 130)

[3] Ali-akbar Agha-mohammadi, Eric Heiden, Karol Hausman, and Gaurav Sukhatme. Confidence-rich grid mapping. *The International Journal of Robotics Research*, 38(12-13):1352–1374, October 2019. ISSN 0278-3649. doi: 10.1177/0278364919839762. (see page: 16)

[4] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974. ISSN 1558-2523. doi: 10.1109/TAC.1974.1100705. (see page: 14)

[5] John Amanatides and Andrew Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *EG 1987-Technical Papers*. Eurographics Association, 1987. doi: 10.2312/egtp.19871000. (see page: 22)

[6] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, USA, January 2007.

Society for Industrial and Applied Mathematics. ISBN 978-0-89871-624-5. (see pages: 50, 57)

[7] Hagai Attias. A variational Bayesian framework for graphical models. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 209–215, Cambridge, MA, USA, November 1999. MIT Press. (see page: 14)

[8] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018. (see page: 130)

[9] Graeme Best, Oliver M. Cliff, Timothy Patten, Ramgopal R. Mettu, and Robert Fitch. Decentralised Monte Carlo Tree Search for Active Perception. In Ken Goldberg, Pieter Abbeel, Kostas Bekris, and Lauren Miller, editors, *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, Springer Proceedings in Advanced Robotics, pages 864–879. Springer International Publishing, Cham, 2020. ISBN 978-3-030-43089-4. doi: 10.1007/978-3-030-43089-4_55. (see page: 9)

[10] Graeme Best, Rohit Garg, John Keller, Geoffrey A. Hollinger, and Sebastian Scherer. Resilient Multi-Sensor Exploration of Multifarious Environments with a Team of Aerial Robots. In *Robotics: Science and Systems XVIII*, volume 18, June 2022. ISBN 978-0-9923747-8-5. (see page: 2)

[11] P. Biber and W. Strasser. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, pages 2743–2748 vol.3, October 2003. doi: 10.1109/IROS.2003.1249285. (see page: 131)

[12] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International computer science institute*, 4(510):126, 1998. (see page: 50)

[13] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon path planning for 3D exploration and surface inspection. *Autonomous Robots*, 42(2):291–306, February 2018. ISSN 1573-7527. doi: 10.1007/s10514-016-9610-0. (see pages: 9, 10)

[14] Christopher M. Bishop. Pattern recognition and machine learning. Information Science and Statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2. (see page: 72)

[15] Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006. (see pages: 18, 20)

[16] Jose Luis Blanco and Pranjal Kumar Rai. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. `https://github.com/jlblancoc/nanoflann`, 2014. (see pages: 22, 139)

[17] David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, March 2006. ISSN 1936-0975, 1931-6690. doi: 10.1214/06-BA104. (see page: 14)

[18] Dorit Borrmann and Andreas Nüchter. Robotic 3D Scan Repository, 2016. (see page: 6)

[19] George EP Box and Mervin E Muller. A note on the generation of random normal deviates. *The annals of mathematical statistics*, 29(2):610–611, 1958. (see page: 55)

[20] R. S. Bradley. Paleoclimatology: Reconstructing climates of the quaternary. Elsevier Science & Technology, Saint Louis, 2014. Online: accessed 3 November 2020. (see page: 89)

[21] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1): 1–43, March 2012. ISSN 1943-0698. doi: 10.1109/TCIAIG.2012.2186810. (see pages: 10, 28)

[22] Adam Bry and Nicholas Roy. Rapidly-exploring Random Belief Trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation*, pages 723–730, Shanghai, China, May 2011. IEEE. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5980508. (see page: 16)

[23] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013. (see page: 140)

[24] Chengtao Cai, Chunsheng Yang, Qidan Zhu, and Yanhua Liang. Collision avoidance in multi-robot systems. In *2007 International Conference on Mechatronics and Automation*, pages 2795–2800. IEEE, 2007. (see page: 95)

[25] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Exploring Large and Complex Environments Fast and Efficiently. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7781–7787, May 2021. doi: 10.1109/ICRA48506.2021.9561916. (see page: 9)

[26] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments. In *Robotics: Science and Systems XVII*, volume 17, July 2021. ISBN 978-0-9923747-7-8. (see page: 10)

[27] Luca Carlone. Estimation Contracts for Outlier-Robust Geometric Perception. *Foundations and Trends® in Robotics*, 11(2-3):90–224, June 2023. ISSN 1935-8253, 1935-8261. doi: 10.1561/2300000077. (see page: 105)

[28] Miguel Á. Carreira-Perpiñán. Fast nonparametric clustering with Gaussian blurring mean-shift. In *Proceedings of the 23rd International Conference on Machine Learning - ICML '06*, pages 153–160, Pittsburgh, Pennsylvania, 2006. ACM Press. ISBN 978-1-59593-383-6. doi: 10.1145/1143844.1143864. (see page: 51)

[29] Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping. In *Robotics: Science and Systems XI*, volume 11, July 2015. ISBN 978-0-9923747-1-6. (see page: 10)

[30] Benjamin Charrow, Sikang Liu, Vijay Kumar, and Nathan Michael. Information-theoretic mapping using Cauchy-Schwarz Quadratic Mutual Information. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4791–4798, May 2015. doi: 10.1109/ICRA.2015.7139865. (see pages: 10, 28, 95, and 98)

[31] Guillaume Chaslot. *Monte-Carlo Tree Search*. PhD thesis, Universiteit Maastricht, 2010. (see pages: 28, 95)

[32] Guillaume M. J. B. Chaslot, Mark H. M. Winands, and H. Jaap van den Herik. Parallel Monte-Carlo Tree Search. In H. Jaap van den Herik, Xinhe Xu, Zongmin Ma, and Mark H. M. Winands, editors, *Computers and Games*, Lecture Notes in Computer Science, pages 60–71, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-87608-3. doi: 10.1007/978-3-540-87608-3_6. (see page: 10)

[33] Timothy Chen, Aiden Swann, Javier Yu, Ola Shorinwa, Riku Murai, Monroe Kennedy III, and Mac Schwager. SAFER-Splat: A Control Barrier Function for Safe Navigation with Online Gaussian Splatting Maps, September 2024. (see page: 15)

[34] Yizhou Chen, Shupeng Lai, Jinqiang Cui, Biao Wang, and Ben M. Chen. GPU-Accelerated Incremental Euclidean Distance Transform for Online Motion Planning of Mobile Robots. *IEEE Robotics and Automation Letters*, 7(3): 6894–6901, July 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3177852. (see page: 15)

[35] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, August 1995. ISSN 1939-3539. doi: 10.1109/34.400568. (see page: 50)

[36] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust Reconstruction of Indoor Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015. (see pages: 62, 84, and 135)

[37] Timothy H Chung, Viktor Orekhov, and Angela Maio. Into the robotic depths: Analysis and insights from the darpa subterranean challenge. *Annual Review of Control, Robotics, and Autonomous Systems*, 6, 2022. (see page: 130)

[38] Titus Cieslewski, Elia Kaufmann, and Davide Scaramuzza. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2135–2142, September 2017. doi: 10.1109/IROS.2017.8206030. (see pages: 9, 10)

[39] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (5):603–619, May 2002. ISSN 01628828. doi: 10.1109/34.1000236. (see pages: 51, 52, and 139)

[40] L. Contin and S. Battista. Performance evaluation of video coding schemes working at very low bit rates. In *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume v, pages V/409–V/412 vol.5, April 1994. doi: 10.1109/ICASSP.1994.389401. (see page: 91)

[41] Micah Corah and Nathan Michael. Distributed Submodular Maximization on Partition Matroids for Planning on Large Sensor Networks. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6792–6799, December 2018. doi: 10.1109/CDC.2018.8619396. (see page: 28)

[42] Micah Corah and Nathan Michael. Volumetric Objectives for Multi-Robot Exploration of Three-Dimensional Environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9043–9050, May 2021. doi: 10.1109/ICRA48506.2021.9561226. (see page: 3)

[43] Micah Corah, Cormac O'Meadhra, Kshitij Goel, and Nathan Michael. Communication-Efficient Planning and Mapping for Multi-Robot Exploration in Large Environments. *IEEE Robotics and Automation Letters*, 4(2): 1715–1721, April 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2897368. (see pages: 8, 9, 10, 14, 28, 32, 49, 50, and 130)

[44] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021. (see page: 133)

[45] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. The heat method for distance computation. *Communications of the ACM*, 60(11):90–99, October 2017. ISSN 0001-0782, 1557-7317. doi: 10.1145/3131280. (see page: 16)

[46] Anna Dai, Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas, and Stefan Leutenegger. Fast Frontier-based Information-driven Autonomous Exploration with an MAV. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9570–9576, May 2020. doi: 10.1109/ICRA40945.2020.9196707. (see pages: 9, 100)

[47] Tung Dang, Marco Tranzatto, Shehryar Khattak, Frank Mascarich, Kostas Alexis, and Marco Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, 37(8):1363–1388, 2020. ISSN 1556-4967. doi: 10.1002/rob.21993. (see page: 2)

[48] Frank Dellaert, V Agrawal, A Jain, M Sklar, and M Xie. Gtsam. https://borg.cc.gatech.edu, 2012. (see page: 138)

[49] Mihir Dharmadhikari, Tung Dang, Lukas Solanka, Johannes Loje, Huan Nguyen, Nikhil Khedekar, and Kostas Alexis. Motion Primitives-based Path Planning for Fast and Agile Exploration using Aerial Robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 179–185, May 2020. doi: 10.1109/ICRA40945.2020.9196964. (see pages: 2, 9, and 100)

[50] Aditya Dhawale and Nathan Michael. Efficient Parametric Multi-Fidelity Surface Mapping. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, July 2020. ISBN 978-0-9923747-6-1. doi: 10.15607/RSS.2020.XVI.073. (see pages: 16, 73, and 132)

[51] Aditya Dhawale, Kumar Shaurya Shankar, and Nathan Michael. Fast Monte-Carlo Localization on Aerial Vehicles Using Approximate Continuous Belief Representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5851–5859, 2018. (see page: 105)

[52] Aditya Dhawale, Xuning Yang, and Nathan Michael. Reactive Collision Avoidance Using Real-Time Local Gaussian Mixture Model Maps. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3545–3550, October 2018. doi: 10.1109/IROS.2018.8593723. (see page: 17)

[53] Kevin Doherty, Jinkun Wang, and Brendan Englot. Bayesian generalized kernel inference for occupancy map prediction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3118–3124, May 2017. doi: 10.1109/ICRA.2017.7989356. (see page: 132)

[54] Kevin Doherty, Tixiao Shan, Jinkun Wang, and Brendan Englot. Learning-Aided 3-D Occupancy Mapping With Bayesian Generalized Kernel Inference. *IEEE Transactions on Robotics*, 35(4):953–966, August 2019. ISSN 1941-0468. doi: 10.1109/TRO.2019.2912487. (see page: 132)

[55] Haolin Dong, Jincheng Yu, Yuanfan Xu, Zhilin Xu, Zhaoyang Shen, Jiahao Tang, Yuan Shen, and Yu Wang. MR-GMMapping: Communication Efficient Multi-Robot Mapping System via Gaussian Mixture Model. *IEEE Robotics*

and Automation Letters, 7(2):3294–3301, April 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3145059. (see page: 13)

[56] Noel E. Du Toit and J. W. Burdick. Probabilistic Collision Checking With Chance Constraints. *IEEE Transactions on Robotics*, 27(4):809–815, August 2011. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2011.2116190. (see page: 16)

[57] Daniel Duberg and Patric Jensfelt. UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown. *IEEE Robotics and Automation Letters*, 5(4):6411–6418, October 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.3013861. (see pages: 12, 132)

[58] Daniel Duberg and Patric Jensfelt. UFOExplorer: Fast and Scalable Sampling-Based Exploration With a Graph-Based Planning Structure. *IEEE Robotics and Automation Letters*, 7(2):2487–2494, April 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3142923. (see page: 9)

[59] Kamak Ebadi, Lukas Bernreiter, Harel Biggie, Gavin Catt, Yun Chang, Arghya Chatterjee, Christopher E. Denniston, Simon-Pierre Deschênes, Kyle Harlow, Shehryar Khattak, Lucas Nogueira, Matteo Palieri, Pavel Petráček, Matěj Petrlík, Andrzej Reinke, Vít Krátký, Shibo Zhao, Ali-akbar Aghamohammadi, Kostas Alexis, Christoffer Heckman, Kasra Khosoussi, Navinda Kottege, Benjamin Morrell, Marco Hutter, Fred Pauling, François Pomerleau, Martin Saska, Sebastian Scherer, Roland Siegwart, Jason L. Williams, and Luca Carlone. Present and Future of SLAM in Extreme Underground Environments, August 2022. (see pages: 1, 3)

[60] B. Eckart, K. Kim, and J. Kautz. HGMR: Hierarchical Gaussian Mixtures for Adaptive 3D Registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 705–721, 2018. (see page: 132)

[61] Ben Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Accelerated Generative Models for 3D Point Cloud Data. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5497–5505, Las Vegas, NV, USA, June 2016. IEEE. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.593. (see pages: 13, 16, 49, 50, 69, and 132)

[62] Benjamin Eckart. *Compact Generative Models of Point Cloud Data for 3D Perception*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, October 2017. (see page: 130)

[63] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. ISSN 1558-0814. doi: 10.1109/2.30720. (see pages: 12, 16, and 96)

[64] Davide Falanga, Suseong Kim, and Davide Scaramuzza. How Fast Is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid. *IEEE Robotics*

and Automation Letters, 4(2):1884–1891, April 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2898117. (see page: 11)

[65] Peter R. Florence, John Carter, Jake Ware, and Russ Tedrake. NanoMap: Fast, Uncertainty-Aware Proximity Queries with Lazy Search Over Local 3D Data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7631–7638, May 2018. doi: 10.1109/ICRA.2018.8463195. (see page: 17)

[66] Gilad Francis, Lionel Ott, and Fabio Ramos. Functional Path Optimisation for Exploration in Continuous Occupancy Maps. In Nancy M. Amato, Greg Hager, Shawna Thomas, and Miguel Torres-Torriti, editors, *Robotics Research*, Springer Proceedings in Advanced Robotics, pages 859–875, Cham, 2020. Springer International Publishing. ISBN 978-3-030-28619-4. doi: 10.1007/978-3-030-28619-4_59. (see page: 16)

[67] Winifred F Frick, Jacob F Pollock, Alan C Hicks, Kate E Langwig, D Scott Reynolds, Gregory G Turner, Calvin M Butchkoski, and Thomas H Kunz. An emerging disease causes regional population collapse of a common north american bat species. *Science*, 329(5992):679–682, 2010. (see page: 89)

[68] 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*, 6(2):3553–3560, April 2021. ISSN 2377-3766. doi: 10.1109/LRA.2021.3061989. (see pages: 12, 132)

[69] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1050–1059. PMLR, June 2016. (see page: 17)

[70] Dasong Gao, Peter Zhi Xuan Li, Vivienne Sze, and Sertac Karaman. GEVO: Memory-Efficient Monocular Visual Odometry Using Gaussians, September 2024. (see page: 104)

[71] Kshitij Goel and Wennie Tabib. Incremental Multimodal Surface Mapping via Self-Organizing Gaussian Mixture Models. *IEEE Robotics and Automation Letters*, 8(12):8358–8365, December 2023. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2023.3327670. (see pages: 4, 6, 17, 73, and 74)

[72] Kshitij Goel and Wennie Tabib. Distance and Collision Probability Estimation from Gaussian Surface Models, April 2024. (see pages: 4, 8)

[73] Kshitij Goel, Micah Corah, Curtis Boirum, and Nathan Michael. Fast Exploration Using Multirotors: Analysis, Planning, and Experimentation. In Genya Ishigami and Kazuya Yoshida, editors, *Field and Service Robotics*, Springer Proceedings in Advanced Robotics, pages 291–305, Singapore, 2021.

Springer. ISBN 9789811594601. doi: 10.1007/978-981-15-9460-1_21. (see pages: 4, 6, 40, 95, and 100)

[74] Kshitij Goel, Wennie Tabib, and Nathan Michael. Rapid and High-Fidelity Subsurface Exploration with Multiple Aerial Robots. In Bruno Siciliano, Cecilia Laschi, and Oussama Khatib, editors, *Experimental Robotics*, Springer Proceedings in Advanced Robotics, pages 436–448, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71151-1. doi: 10.1007/978-3-030-71151-1_39. (see pages: 4, 8, and 15)

[75] Kshitij Goel, Yves Georgy Daoud, Nathan Michael, and Wennie Tabib. Hierarchical Collision Avoidance for Adaptive-Speed Multirotor Teleoperation. In *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 20–27, November 2022. doi: 10.1109/SSRR56537.2022.10018782. (see pages: 4, 6)

[76] Kshitij Goel, Nathan Michael, and Wennie Tabib. Probabilistic Point Cloud Modeling via Self-Organizing Gaussian Mixture Models. *IEEE Robotics and Automation Letters*, 8(5):2526–2533, May 2023. ISSN 2377-3766. doi: 10.1109/LRA.2023.3256923. (see pages: 4, 6, 14, 15, 55, 69, 79, 80, 85, 130, 131, 132, 133, 134, 139, 140, and 141)

[77] Nicola Greggio, Alexandre Bernardino, Cecilia Laschi, Paolo Dario, and José Santos-Victor. Fast estimation of Gaussian mixture models for image segmentation. *Machine Vision and Applications*, 23(4):773–789, July 2012. ISSN 1432-1769. doi: 10.1007/s00138-011-0320-5. (see page: 13)

[78] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. (see pages: 77, 134)

[79] Hanzhong Guo, Cheng Lu, Fan Bao, Tianyu Pang, Shuicheng Yan, Chao Du, and Chongxuan Li. Gaussian Mixture Solvers for Diffusion Models, November 2023. (see page: 105)

[80] Abhishek Halder. On the Parameterized Computation of Minimum Volume Outer Ellipsoid of Minkowski Sum of Ellipsoids. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4040–4045, Miami Beach, FL, December 2018. IEEE. ISBN 978-1-5386-1395-5. doi: 10.1109/CDC.2018.8619508. (see pages: 72, 76)

[81] Luxin Han, Fei Gao, Boyu Zhou, and Shaojie Shen. FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4423–4430, November 2019. doi: 10.1109/IROS40897.2019.8968199. (see page: 15)

[82] Mark H Hansen and Bin Yu. Model Selection and the Principle of Minimum Description Length. *Journal of the American Statistical Association*, 96(454):

746–774, June 2001. ISSN 0162-1459. doi: 10.1198/016214501753168398. (see page: 14)

[83] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. (see page: 134)

[84] Søren Hauberg. Directional Statistics with the Spherical Normal Distribution. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 704–711, July 2018. doi: 10.23919/ICIF.2018.8455242. (see page: 105)

[85] Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Jan Kautz, Changliu Liu, Guanya Shi, Xiaolong Wang, Linxi Fan, and Yuke Zhu. HOVER: Versatile Neural Whole-Body Controller for Humanoid Robots, October 2024. (see page: 104)

[86] Eric Heiden, Karol Hausman, Gaurav S. Sukhatme, and Ali-akbar Aghamohammadi. Planning high-speed safe trajectories in confidence-rich maps. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2880–2886, September 2017. doi: 10.1109/IROS.2017.8206120. (see page: 16)

[87] John C. Hempel and Annette Fregeau-Conover. On Call: A Complete Reference for Cave Rescue. National Speleological Society, 2001. ISBN 978-1-879961-16-6. (see page: 1)

[88] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, April 2013. ISSN 1573-7527. doi: 10.1007/s10514-012-9321-0. (see pages: 11, 12, 16, 55, 62, 68, 96, 100, 103, 105, and 131)

[89] Bert Hubert, Thomas Graf, Greg Maxwell, Remco van Mook, Martijn van Oosterhout, P Schroeder, Jasper Spaans, and Pedro Larroy. Linux advanced routing & traffic control. In *Ottawa Linux Symposium*, volume 213. sn, 2002. (see page: 102)

[90] Jeffrey Ichnowski and Ron Alterovitz. Concurrent Nearest-Neighbor Searching for Parallel Sampling-Based Motion Planning in SO(3), SE(3), and Euclidean Spaces. In Marco Morales, Lydia Tapia, Gildardo Sánchez-Ante, and Seth Hutchinson, editors, *Algorithmic Foundations of Robotics XIII*, Springer

Proceedings in Advanced Robotics, pages 69–85, Cham, 2020. Springer International Publishing. ISBN 978-3-030-44051-0. doi: 10.1007/978-3-030-44051-0_5. (see pages: 74, 77)

[91] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. https://github.com/pybind/pybind11. (see page: 134)

[92] Jennifer Jang and Heinrich Jiang. MeanShift++: Extremely Fast Mode-Seeking With Applications to Segmentation and Object Tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4100–4111, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.00409. (see page: 51)

[93] Lucas Janson, Tommy Hu, and Marco Pavone. Safe Motion Planning in Unknown Environments: Optimality Benchmarks and Tractable Policies. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018. ISBN 978-0-9923747-4-7. doi: 10.15607/RSS.2018.XIV.061. (see pages: 2, 3, 11, 24, and 40)

[94] Hongcheng Ji, Haibo Xie, Cheng Wang, and Huayong Yang. E-RRT*: Path Planning for Hyper-Redundant Manipulators. *IEEE Robotics and Automation Letters*, 8(12):8128–8135, December 2023. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2023.3325716. (see page: 16)

[95] Rui Jin, Yuman Gao, Haojian Lu, and Fei Gao. GS-Planner: A Gaussian-Splatting-based Planning Framework for Active High-Fidelity Reconstruction, May 2024. (see page: 15)

[96] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182–193. SPIE, July 1997. doi: 10.1117/12.280797. (see page: 17)

[97] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat Track & Map 3D Gaussians for Dense RGB-D SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024. (see page: 15)

[98] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, July 2023. ISSN 0730-0301. doi: 10.1145/3592433. (see pages: 15, 16, and 73)

[99] Leonid Keselman and Martial Hebert. Approximate Differentiable Rendering with Algebraic Surfaces. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*,

pages 596–614, Berlin, Heidelberg, October 2022. Springer-Verlag. ISBN 978-3-031-19823-6. doi: 10.1007/978-3-031-19824-3_35. (see pages: 15, 16, and 73)

[100] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything, April 2023. (see page: 104)

[101] Steven Kisseleff, Ian F Akyildiz, and Wolfgang H Gerstacker. Survey on advances in magnetic induction-based wireless underground sensor networks. *IEEE Internet of Things Journal*, 5(6):4843–4856, 2018. (see page: 96)

[102] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9): 1464–1480, September 1990. ISSN 1558-2256. doi: 10.1109/5.58325. (see page: 14)

[103] Miroslav Kulich, Jan Faigl, and Libor Přeučil. On distance utility in the exploration task. In *2011 IEEE International Conference on Robotics and Automation*, pages 4455–4460, May 2011. doi: 10.1109/ICRA.2011.5980221. (see page: 28)

[104] Johann Laconte, Christophe Debain, Roland Chapuis, François Pomerleau, and Romuald Aufrère. Lambda-Field: A Continuous Counterpart of the Bayesian Occupancy Grid for Risk Assessment. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 167–172, November 2019. doi: 10.1109/IROS40897.2019.8968100. (see page: 16)

[105] Jean-Claude Latombe. Robot Motion Planning. Springer US, Boston, MA, 1991. ISBN 978-0-7923-9206-4 978-1-4615-4022-9. doi: 10.1007/978-1-4615-4022-9. (see page: 16)

[106] Mikko Lauri and Risto Ritala. Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83:15–31, September 2016. ISSN 0921-8890. doi: 10.1016/j.robot.2016.06.008. (see page: 28)

[107] Cedric Le Gentil, Othmane-Latif Ouabi, Lan Wu, Cedric Pradalier, and Teresa Vidal-Calleja. Accurate Gaussian-Process-Based Distance Fields With Applications to Echolocation and Mapping. *IEEE Robotics and Automation Letters*, 9(2):1365–1372, February 2024. ISSN 2377-3766. doi: 10.1109/LRA.2023.3346759. (see pages: 15, 16, 17, 77, 80, 81, 83, 84, 85, and 87)

[108] Bhoram Lee, Clark Zhang, Zonghao Huang, and Daniel D. Lee. Online Continuous Mapping using Gaussian Process Implicit Surfaces. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6884–6890, May 2019. doi: 10.1109/ICRA.2019.8794324. (see pages: 15, 16)

[109] Joshua Levin, Aditya Paranjape, and Meyer Nahon. Motion Planning for a Small Aerobatic Fixed-Wing Unmanned Aerial Vehicle. In *2018 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 8464–8470, October 2018. doi: 10.1109/IROS.2018.8593670. (see page: 2)

[110] Peter Zhi Xuan Li, Sertac Karaman, and Vivienne Sze. Memory-Efficient Gaussian Fitting for Depth Images in Real Time. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8003–8009, May 2022. doi: 10.1109/ICRA46639.2022.9811682. (see pages: 16, 73)

[111] Peter Zhi Xuan Li, Sertac Karaman, and Vivienne Sze. GMMap: Memory-Efficient Continuous Occupancy Map Using Gaussian Mixture Model. *IEEE Transactions on Robotics*, 40:1339–1355, 2024. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2023.3348305. (see pages: 16, 17, and 78)

[112] Anhua Lin and Shih-Ping Han. On the Distance between Two Ellipsoids. *SIAM Journal on Optimization*, 13(1):298–308, January 2002. ISSN 1052-6234, 1095-7189. doi: 10.1137/S1052623401396510. (see page: 73)

[113] Haohan Lin, Xuzhan Chen, Matthew Tucsok, Li Ji, and Homayoun Najjaran. Evaluating Initialization Methods for Discriminative and Fast-Converging HGMM Point Clouds. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13871–13876, May 2021. doi: 10.1109/ICRA48506.2021.9560882. (see page: 13)

[114] Sikang Liu, Kartik Mohta, Nikolay Atanasov, and Vijay Kumar. Search-Based Motion Planning for Aggressive Flight in SE(3). *IEEE Robotics and Automation Letters*, 3(3):2439–2446, July 2018. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2018.2795654. (see pages: 16, 17)

[115] Tianyu Liu, Fu Zhang, Fei Gao, and Jia Pan. Tight Collision Probability for UAV Motion Planning in Uncertain Environment. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1055–1062, Detroit, MI, USA, October 2023. IEEE. ISBN 978-1-66549-190-7. doi: 10.1109/IROS55552.2023.10342141. (see pages: 16, 76, and 83)

[116] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979. ISSN 0001-0782, 1557-7317. doi: 10.1145/359156.359164. (see page: 17)

[117] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022. doi: 10.1126/scirobotics.abm6074. (see page: 133)

[118] Martin Magnusson, Achim Lilienthal, and Tom Duckett. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, 2007. ISSN 1556-4967. doi: 10.1002/rob.20204. (see pages: 12, 55, and 131)

[119] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, September 2012. ISSN 1558-223X. doi: 10.1109/MRA.2012.2206474. (see page: 24)

[120] M. Masry and S. S. Hemami. An analysis of subjective quality in low bit rate video. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, volume 1, pages 465–468 vol.1, Oct 2001. doi: 10.1109/ICIP.2001.959054. (see page: 91)

[121] A. M. Mathai and S. B. Provost. Quadratic Forms in Random Variables: Theory and Applications. Marcel Dek ker, 1992. (see page: 76)

[122] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. (see page: 14)

[123] Geoffrey J. McLachlan and Suren Rathnayake. On the number of components in a Gaussian mixture model: Number of components in a Gaussian mixture model. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(5):341–355, September 2014. ISSN 19424787. doi: 10.1002/widm.1135. (see page: 13)

[124] Guofeng Mei, Fabio Poiesi, Cristiano Saltori, Jian Zhang, Elisa Ricci, and Nicu Sebe. Overlap-guided Gaussian Mixture Models for Point Cloud Registration. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4500–4509, Waikoloa, HI, USA, January 2023. IEEE. ISBN 978-1-66549-346-8. doi: 10.1109/WACV56688.2023.00449. (see page: 105)

[125] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011. doi: 10.1109/ICRA.2011.5980409. (see pages: 2, 26)

[126] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro-UAV Testbed. *IEEE Robotics & Automation Magazine*, 17(3):56–65, September 2010. ISSN 1558-223X. doi: 10.1109/MRA.2010.937855. (see page: 24)

[127] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 405–421, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8. doi: 10.1007/978-3-030-58452-8_24. (see pages: 13, 14, and 17)

[128] Alexander Millane, Helen Oleynikova, Emilie Wirbel, Remo Steiner, Vikram Ramasamy, David Tingdahl, and Roland Siegwart. Nvblox: GPU-Accelerated Incremental Signed Distance Field Mapping, March 2024. (see page: 103)

[129] Anastasios I. Mourikis and Stergios I. Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007. doi: 10.1109/ROBOT.2007.364024. (see page: 72)

[130] Ken Museth. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics*, 32(3):1–22, June 2013. ISSN 0730-0301, 1557-7368. doi: 10.1145/2487228.2487235. (see page: 74)

[131] Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. OpenVDB: An open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, page 1, New York, NY, USA, July 2013. Association for Computing Machinery. ISBN 978-1-4503-2339-0. doi: 10.1145/2504435.2504454. (see page: 132)

[132] NASA. Mars science laboratory data rates/returns, 2019. (see page: 96)

[133] Javier Navarrete, Diego Viejo, and Miguel Cazorla. Compression and registration of 3D point clouds using GMMs. *Pattern Recognition Letters*, 110:8–15, July 2018. ISSN 0167-8655. doi: 10.1016/j.patrec.2018.03.017. (see page: 13)

[134] Erik Nelson, Micah Corah, and Nathan Michael. Environment model adaptation for mobile robot exploration. *Autonomous Robots*, 42(2):257–272, February 2018. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-017-9669-2. (see pages: 2, 11)

[135] Huan Nguyen, Sondre Holm Fyhn, Paolo De Petris, and Kostas Alexis. Motion Primitives-based Navigation Planning using Deep Collision Prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9660–9667, Philadelphia, PA, USA, May 2022. IEEE. ISBN 978-1-72819-681-7. doi: 10.1109/ICRA46639.2022.9812231. (see page: 17)

[136] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, January 2012. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364911421039. (see page: 13)

[137] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, September 2017. doi: 10.1109/IROS.2017.8202315. (see pages: 12, 15, 16, 62, 105, and 131)

[138] Cormac O'Meadhra, Wennie Tabib, and Nathan Michael. Variable Resolution Occupancy Mapping Using Gaussian Mixture Models. *IEEE Robotics and Automation Letters*, 4(2):2015–2022, April 2019. ISSN 2377-3766. doi: 10.1109/LRA.2018.2889348. (see pages: 8, 13, 16, 20, 69, and 98)

[139] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022. (see page: 14)

[140] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. iSDF: Real-Time Neural Signed Distance Fields for Robot Perception. In *Robotics: Science and Systems XVIII*, volume 18, June 2022. ISBN 978-0-9923747-8-5. (see pages: 17, 79)

[141] Christos Papachristos, Shehryar Khattak, and Kostas Alexis. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575, May 2017. doi: 10.1109/ICRA.2017.7989531. (see page: 9)

[142] Aditya A. Paranjape, Kevin C. Meier, Xichen Shi, Soon-Jo Chung, and Seth Hutchinson. Motion primitives and 3D path planning for fast flight through a forest. *The International Journal of Robotics Research*, 34(3):357–377, March 2015. ISSN 0278-3649. doi: 10.1177/0278364914558017. (see page: 2)

[143] Kevin Joseph Patrick. Pennsylvania caves and other rocky roadside wonders. Stackpole Books, 2004. (see page: 89)

[144] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011. (see page: 139)

[145] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. (see page: 140)

[146] Mihail Pivtoraiko, Issa A.D. Nesnas, and Alonzo Kelly. Autonomous robot navigation using advanced motion primitives. In *2009 IEEE Aerospace Conference*, pages 1–7, March 2009. doi: 10.1109/AERO.2009.4839309. (see pages: 1, 40)

[147] J. C. Principe. Information theoretic learning: Renyi's entropy and kernel perspectives. Information Science and Statistics. Springer, New York, 2010. ISBN 978-1-4419-1569-6. (see pages: 6, 50)

[148] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34 (4):1004–1020, August 2018. ISSN 1941-0468. doi: 10.1109/TRO.2018.2853729. (see page: 32)

[149] Lun Quan, Zhiwei Zhang, Xingguang Zhong, Chao Xu, and Fei Gao. EVA-Planner: Environmental Adaptive Quadrotor Planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 398–404, May 2021. doi: 10.1109/ICRA48506.2021.9561759. (see pages: 2, 11)

[150] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009. (see page: 133)

[151] Fabio Ramos and Lionel Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14):1717–1730, December 2016. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364916684382. (see pages: 13, 16)

[152] Sudhir Rao, Allan de Medeiros Martins, and José C. Príncipe. Mean shift: An information theoretic perspective. *Pattern Recognition Letters*, 30(3):222–230, February 2009. ISSN 0167-8655. doi: 10.1016/j.patrec.2008.09.011. (see page: 50)

[153] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian Processes for Machine Learning. The MIT Press, November 2005. ISBN 978-0-262-25683-4. doi: 10.7551/mitpress/3206.001.0001. (see page: 16)

[154] Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494, May 2009. doi: 10.1109/ROBOT.2009.5152817. (see page: 15)

[155] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps. *IEEE Robotics and Automation Letters*, 5(1):227–234, January 2020. ISSN 2377-3766. doi: 10.1109/LRA.2019.2953859. (see page: 132)

[156] Victor Reijgwart, Cesar Cadena, Roland Siegwart, and Lionel Ott. Efficient volumetric mapping of multi-scale environments using wavelet-based compression. In *Robotics: Science and Systems XIX*, volume 19, July 2023. ISBN 978-0-9923747-9-2. (see pages: 105, 132)

[157] Elon Rimon and Stephen P. Boyd. Obstacle Collision Detection Using Best Ellipsoid Fit. *Journal of Intelligent and Robotic Systems*, 18(2):105–126,

February 1997. ISSN 1573-0409. doi: 10.1023/A:1007960531949. (see pages: 72, 73, 74, and 76)

[158] Sipu Ruan, Karen L. Poblete, Hongtao Wu, Qianli Ma, and Gregory S. Chirikjian. Efficient Path Planning in Narrow Passages for Robots With Ellipsoidal Components. *IEEE Transactions on Robotics*, 39(1):110–127, February 2023. ISSN 1941-0468. doi: 10.1109/TRO.2022.3187818. (see page: 16)

[159] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, Juha Ala-Luhtala, and Achim J. Lilienthal. Normal Distributions Transform Occupancy Maps: Application to large-scale online 3D mapping. In *2013 IEEE International Conference on Robotics and Automation*, pages 2233–2238, May 2013. doi: 10.1109/ICRA.2013.6630878. (see page: 131)

[160] Jari P. Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim J. Lilienthal. 3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments. *The International Journal of Robotics Research*, 32(14):1627–1644, December 2013. ISSN 0278-3649. doi: 10.1177/0278364913499415. (see page: 14)

[161] Kelsey Saulnier, Nikolay Atanasov, George J. Pappas, and Vijay Kumar. Information Theoretic Active Exploration in Signed Distance Fields. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4080–4085, May 2020. doi: 10.1109/ICRA40945.2020.9196882. (see page: 15)

[162] Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J. Pappas. Anytime Planning for Decentralized Multirobot Active Information Gathering. *IEEE Robotics and Automation Letters*, 3(2):1025–1032, April 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2794608. (see page: 10)

[163] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978. ISSN 0090-5364. (see page: 14)

[164] J A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591–1595, February 1996. ISSN 0027-8424. (see page: 39)

[165] Alex Spitzer, Xuning Yang, John Yao, Aditya Dhawale, Kshitij Goel, Mosam Dabhi, Matt Collins, Curtis Boirum, and Nathan Michael. Fast and Agile Vision-Based Flight with Teleoperation and Collision Avoidance on a Multirotor. In Jing Xiao, Torsten Kröger, and Oussama Khatib, editors, *Proceedings of the 2018 International Symposium on Experimental Robotics*, Springer Proceedings in Advanced Robotics, pages 524–535, Cham, 2020. Springer International Publishing. ISBN 978-3-030-33950-0. doi: 10.1007/978-3-030-33950-0_45. (see pages: 10, 26, 27, 32, and 40)

[166] Shobhit Srivastava. Efficient, multi-fidelity perceptual representations via hierarchical gaussian mixture models. Master's thesis, Carnegie Mellon

University, Pittsburgh PA, August 2017. (see pages: 21, 49, 50, 53, 62, 63, and 66)

[167] Shobhit Srivastava and Nathan Michael. Efficient, Multifidelity Perceptual Representations via Hierarchical Gaussian Mixture Models. *IEEE Transactions on Robotics*, 35(1):248–260, February 2019. ISSN 1941-0468. doi: 10.1109/TRO.2018.2878363. (see pages: 13, 16, 17, 53, 56, 66, 69, 73, and 132)

[168] V. Stamenković, L. W. Beegle, K. Zacny, D. D. Arumugam, P. Baglioni, N. Barba, J. Baross, M. S. Bell, R. Bhartia, J. G. Blank, P. J. Boston, D. Breuer, W. Brinckerhoff, M. S. Burgin, I. Cooper, V. Cormarkovic, A. Davila, R. M. Davis, C. Edwards, G. Etiope, W. W. Fischer, D. P. Glavin, R. E. Grimm, F. Inagaki, J. L. Kirschvink, A. Kobayashi, T. Komarek, M. Malaska, J. Michalski, B. Ménez, M. Mischna, D. Moser, J. Mustard, T. C. Onstott, V. J. Orphan, M. R. Osburn, J. Plaut, A.-C. Plesa, N. Putzig, K. L. Rogers, L. Rothschild, M. Russell, H. Sapers, B. Sherwood Lollar, T. Spohn, J. D. Tarnas, M. Tuite, D. Viola, L. M. Ward, B. Wilcox, and R. Woolley. The next frontier for planetary and human exploration. *Nature Astronomy*, 3(2):116–120, February 2019. ISSN 2397-3366. doi: 10.1038/s41550-018-0676-9. (see page: 1)

[169] Johannes Stephan, Ole Pfeifle, Stefan Notter, Federico Pinchetti, and Walter Fichter. Precise Tracking of Extended Three-Dimensional Dubins Paths for Fixed-Wing Aircraft. *Journal of Guidance, Control, and Dynamics*, 43(12):2399–2405, 2020. ISSN 0731-5090. doi: 10.2514/1.G005240. (see page: 2)

[170] G. W. Stewart. The Efficient Generation of Random Orthogonal Matrices with an Application to Condition Estimators. *SIAM Journal on Numerical Analysis*, 17(3):403–409, 1980. ISSN 0036-1429. (see page: 78)

[171] Todor Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim J Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *The International Journal of Robotics Research*, 31(12):1377–1393, 2012. (see page: 131)

[172] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021. (see page: 14)

[173] Shuo Sun, Malcolm Mielle, Achim J. Lilienthal, and Martin Magnusson. High-Fidelity SLAM Using Gaussian Splatting with Rendering-Guided Densification and Regularized Optimization, October 2024. (see page: 15)

[174] Hsi Guang Sung. *Gaussian Mixture Regression and Classification*. PhD thesis, Rice University, Houston, Texas, May 2004. (see pages: 50, 56)

[175] Wennie Tabib. *Approximate Continuous Belief Distributions for Exploration*. CMU-CS-TR-19-108, Carnegie Mellon University, Pittsburgh, PA, USA, May 2019. (see page: 140)

[176] Wennie Tabib and Nathan Michael. Simultaneous Localization and Mapping of Subterranean Voids with Gaussian Mixture Models. In Genya Ishigami and Kazuya Yoshida, editors, *Field and Service Robotics*, pages 173–187, Singapore, 2021. Springer. ISBN 9789811594601. doi: 10.1007/978-981-15-9460-1_13. (see pages: 15, 20, 104, 130, 131, 133, 137, and 141)

[177] Wennie Tabib, Red Whittaker, and Nathan Michael. Efficient multi-sensor exploration using dependent observations and conditional mutual information. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 42–47, October 2016. doi: 10.1109/SSRR.2016.7784275. (see pages: 2, 12)

[178] Wennie Tabib, Cormac O'Meadhra, and Nathan Michael. On-Manifold GMM Registration. *IEEE Robotics and Automation Letters*, 3(4):3805–3812, October 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2856279. (see pages: 15, 104, 130, 131, 133, 137, and 141)

[179] Wennie Tabib, Kshitij Goel, John Yao, Mosam Dabhi, Curtis Boirum, and Nathan Michael. Real-Time Information-Theoretic Exploration with Gaussian Mixture Model Maps. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. ISBN 978-0-9923747-5-4. doi: 10.15607/RSS.2019.XV.061. (see pages: 8, 10, 14, 16, 18, 20, 22, 50, and 98)

[180] Wennie Tabib, Kshitij Goel, John Yao, Curtis Boirum, and Nathan Michael. Autonomous Cave Surveying With an Aerial Robot. *IEEE Transactions on Robotics*, 38(2):1016–1032, April 2022. ISSN 1941-0468. doi: 10.1109/TRO.2021.3104459. (see pages: 4, 8, 11, 18, 22, 41, 55, 62, 66, 69, 88, 92, 94, 99, 100, 130, 133, and 141)

[181] Yuezhan Tao, Dexter Ong, Varun Murali, Igor Spasojevic, Pratik Chaudhari, and Vijay Kumar. RT-GuIDE: Real-Time Gaussian splatting for Information-Driven Exploration, September 2024. (see page: 15)

[182] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. (see page: 133)

[183] Antony Thomas, Fulvio Mastrogiovanni, and Marco Baglietto. Exact and Bounded Collision Probability for Motion Planning Under Gaussian Uncertainty. *IEEE Robotics and Automation Letters*, 7(1):167–174, January 2022. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2021.3121073. (see pages: 76, 77)

[184] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3): 52–57, March 2002. ISSN 0001-0782, 1557-7317. doi: 10.1145/504729.504754. (see pages: 21, 22, and 39)

[185] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P. How, and Luca Carlone. Kimera-Multi: Robust, Distributed, Dense Metric-Semantic SLAM for Multi-Robot Systems. *IEEE Transactions on Robotics*, 38(4):2022–2038, August 2022. ISSN 1941-0468. doi: 10.1109/TRO.2021.3137751. (see page: 104)

[186] Jesus Tordesillas, Brett T. Lopez, Michael Everett, and Jonathan P. How. FASTER: Fast and Safe Trajectory Planner for Navigation in Unknown Environments. *IEEE Transactions on Robotics*, 38(2):922–938, April 2022. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2021.3100142. (see pages: 11, 15)

[187] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. Differentiable Collision Detection for a Set of Convex Primitives. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3663–3670, London, United Kingdom, May 2023. IEEE. ISBN 9798350323658. doi: 10.1109/ICRA48891.2023.10160716. (see page: 73)

[188] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. (see page: 105)

[189] J. J. Verbeek, N. Vlassis, and B. J. A. Kröse. Self-organizing mixture models. *Neurocomputing*, 63:99–123, January 2005. ISSN 0925-2312. doi: 10.1016/j.neucom.2004.04.008. (see page: 14)

[190] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul H. J. Kelly, and Stefan Leutenegger. Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151, April 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2792537. (see page: 132)

[191] Red Whittaker et al. Exploration of planetary skylights and tunnels, 2014. (see page: 96)

[192] Amy Williams, Steve Barrus, R Keith Morley, and Peter Shirley. An efficient and robust ray-box intersection algorithm. In *ACM SIGGRAPH 2005 Courses*, pages 9–es. Association for Computing Machinery, 2005. (see page: 22)

[193] Oliver Williams and Andrew Fitzgibbon. Gaussian Process Implicit Surfaces. Technical report, Microsoft, 2007. (see page: 16)

[194] Lan Wu, Ki Myung Brian Lee, Liyang Liu, and Teresa Vidal-Calleja. Faithful Euclidean Distance Field From Log-Gaussian Process Implicit Surfaces. *IEEE*

Robotics and Automation Letters, 6(2):2461–2468, April 2021. ISSN 2377-3766. doi: 10.1109/LRA.2021.3061356. (see page: 16)

[195] Lan Wu, Cedric Le Gentil, and Teresa Vidal-Calleja. Pseudo Inputs Optimisation for Efficient Gaussian Process Distance Fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7249–7255, October 2023. doi: 10.1109/IROS55552.2023.10342483. (see page: 16)

[196] Lan Wu, Ki Myung Brian Lee, Cedric Le Gentil, and Teresa Vidal-Calleja. Log-GPIS-MOP: A Unified Representation for Mapping, Odometry, and Planning. *IEEE Transactions on Robotics*, 39(5):4078–4094, October 2023. ISSN 1941-0468. doi: 10.1109/TRO.2023.3296982. (see pages: 15, 84)

[197] Zijun Xu, Rui Jin, Ke Wu, Yi Zhao, Zhiwei Zhang, Jieru Zhao, Zhongxue Gan, and Wenchao Ding. HGS-Planner: Hierarchical Planning Framework for Active Scene Reconstruction Using 3D Gaussian Splatting, September 2024. (see page: 15)

[198] Ryoya Yamasaki. *Convergence Analysis of Mean Shift Type Algorithms*. Doctoral thesis, Kyoto University, March 2024. (see page: 105)

[199] Ryoya Yamasaki and Toshiyuki Tanaka. Convergence Analysis of Blurring Mean Shift, February 2024. (see page: 105)

[200] Ryoya Yamasaki and Toshiyuki Tanaka. Convergence Analysis of Mean Shift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10): 6688–6698, October 2024. ISSN 1939-3539. doi: 10.1109/TPAMI.2024.3385920. (see page: 105)

[201] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous Agents - AGENTS '98*, pages 47–53, Minneapolis, Minnesota, United States, 1998. ACM Press. ISBN 978-0-89791-983-8. doi: 10.1145/280765.280773. (see page: 9)

[202] Zike Yan, Xin Wang, and Hongbin Zha. Online Learning of a Probabilistic and Adaptive Scene Representation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13106–13116, June 2021. doi: 10.1109/CVPR46437.2021.01291. (see pages: 13, 55)

[203] Xuning Yang, Koushil Sreenath, and Nathan Michael. A framework for efficient teleoperation via online adaptation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5948–5953, May 2017. doi: 10.1109/ICRA.2017.7989701. (see pages: 10, 26, 39, and 40)

[204] Xuning Yang, Ayush Agrawal, Koushil Sreenath, and Nathan Michael. Online adaptive teleoperation via motion primitives for mobile robots. *Autonomous Robots*, 43(6):1357–1373, August 2019. ISSN 1573-7527. doi: 10.1007/s10514-018-9753-2. (see pages: 2, 104)

[205] Ji Zhang, Chen Hu, Rushat Gupta Chadha, and Sanjiv Singh. Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, 37(8):1300–1313, 2020. ISSN 1556-4967. doi: 10.1002/rob.21952. (see page: 11)

[206] Yikun Zhang and Yen-Chi Chen. Mode and Ridge Estimation in Euclidean and Directional Product Spaces: A Mean Shift Approach, October 2021. (see page: 105)

[207] Zhengdong Zhang, Theia Henderson, Sertac Karaman, and Vivienne Sze. FSMI: Fast computation of Shannon mutual information for information-theoretic mapping. *The International Journal of Robotics Research*, 39(9):1155–1177, August 2020. ISSN 0278-3649. doi: 10.1177/0278364920921941. (see pages: 10, 98, and 102)

[208] Xingguang Zhong, Yue Pan, Jens Behley, and Cyrill Stachniss. SHINE-Mapping: Large-Scale 3D Mapping Using Sparse Hierarchical Implicit Neural Representations. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8371–8377, May 2023. doi: 10.1109/ICRA48891.2023.10160907. (see page: 14)

[209] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. FUEL: Fast UAV Exploration Using Incremental Frontier Structure and Hierarchical Planning. *IEEE Robotics and Automation Letters*, 6(2):779–786, April 2021. ISSN 2377-3766. doi: 10.1109/LRA.2021.3051563. (see page: 9)

[210] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4):1–8, July 2013. ISSN 0730-0301, 1557-7368. doi: 10.1145/2461912.2461919. (see pages: 55, 59, 60, 62, and 84)

[211] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing, January 2018. (see pages: 68, 133, and 134)

[212] Hai Zhu and Javier Alonso-Mora. Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments. *IEEE Robotics and Automation Letters*, 4 (2):776–783, April 2019. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2019.2893494. (see pages: 16, 77, and 83)

[213] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12776–12786, New Orleans, LA, USA, June 2022. IEEE. ISBN 978-1-66546-946-3. doi: 10.1109/CVPR52688.2022.01245. (see page: 15)

[214] Ehsan Zobeidi, Alec Koppel, and Nikolay Atanasov. Dense Incremental Metric-Semantic Mapping for Multiagent Systems via Sparse Gaussian Process Regression. *IEEE Transactions on Robotics*, 38(5):3133–3153, October 2022. ISSN 1941-0468. doi: 10.1109/TRO.2022.3168733. (see pages: 2, 13)

Appendices

APPENDIX A

GIRA: Gaussian Mixture Models for Inference and Robot Autonomy

To navigate in and interact with the world, robots acquire, assimilate, and respond to sensor data. Models that enable perception in the large and small [8] are amenable to diverse robotics applications and have the potential to drastically increase robotic capabilities while addressing limitations in the way complex perception systems are developed today.

Recent large-scale robotic reconstruction deployments, like the DARPA Subterranean (Sub-T) Challenge [37], have highlighted the need for map compression to increase the reconstruction rate, an example of perception in the large, by facilitating information sharing. Further, state-of-the-art perception systems typically leverage separate concurrent perceptual processing pipelines, which increases computation, redundancy, and complexity [62]. For example, the highly sophisticated perception module of the NeBula system architecture [2] processes the same LiDAR data repeatedly (e.g., odometry, SLAM, terrain mapping, etc.), which is inefficient. Instead, what is needed is a unified framework for common perceptual processing elements, which is compact, generative, and amenable for deployment on low-power embedded systems [62].

Gaussian mixture models (GMMs) provide high-fidelity and communication-efficient point cloud modeling and inference [43] in real-world environments [180]. Recent works have demonstrated precise, high-fidelity representation of fine details required for perception in the small [76]. However, there are few open-source implementations, which poses a barrier to broad adoption by the general robotics community. To bridge this gap, this thesis introduces GIRA, an open-source, unified framework (Fig. A.1) for point cloud modeling, occupancy modeling, and pose estimation using GMMs based on [178, 176, 180]. In addition, GIRA includes a novel systems contribution, which consists of GPU-accelerated functions to learn GMMs 10-100x faster compared to existing CPU implementations. The software and associated datasets are open-sourced[1] to accelerate innovation and adoption of these techniques.

A.1 Prior Open-Source Perception Frameworks

This section reviews open-source perception frameworks for compact, high-resolution point cloud modeling, pose estimation, and occupancy modeling for robotics applica-

[1]Project webpage: https://github.com/gira3d

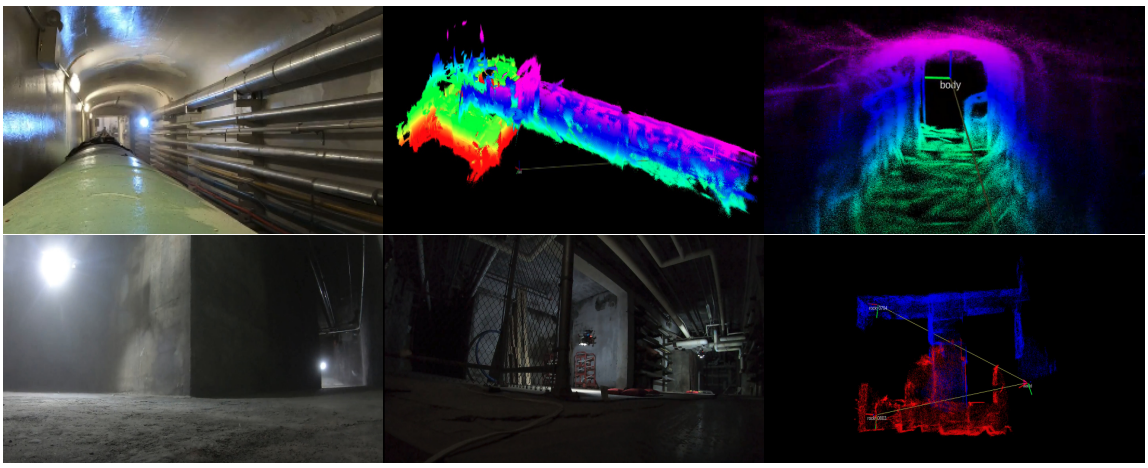


Figure A.1: GIRA has been deployed on size, weight, and power constrained aerial systems in real-world and unstructured environments. (Top left) A single aerial robot flies through an industrial tunnel and (top center) generates a high-fidelity Gaussian mixture model (GMM) map of the environment. (Top right) A close-up view of the reconstructed area around the robot. (Bottom left and bottom center) A team of two robots fly through a dark tunnel environment and produce a (bottom right) map, which is resampled from the underlying GMM and colored red or blue according to which robot took the observation. Videos of these experiments are available at: https://youtu.be/qkbxfxgCoV0 and https://youtu.be/t9iYd33oz3g.

tions. These works are compared and contrasted with GIRA.

The Normal Distributions Transform (NDT)[2] framework was introduced by Biber and Strasser [11] for scan registration and later extended to 3D registration [118] and occupancy modeling [159]. Goel et al. [76] demonstrate that NDTMap provides higher representation fidelity compared to Octomap[3] [88], but at the cost of increased disk storage requirements. While NDTMap provides distribution to distribution registration [171], Octomap does not provide analogous functionality. In contrast to these representations, GIRA provides higher memory-efficiency and surface reconstruction fidelity [76] as well as distribution to distribution registration [178, 176]. Further, NDTMap provides a CPU implementation, while GIRA provides both CPU and GPU implementations for multimodal environment modeling.

Oleynikova et al. [137] develop Voxblox, which uses Truncated Signed Distance Fields (TSDFs), for high-resolution reconstruction and occupancy mapping. The weights for the TSDFs are stored in a coarse fixed-resolution regular grid. Voxblox grows dynamically, but suffers from the same memory-efficiency limitation as the NDTMap. In contrast, the GIRA framework enables high-resolution surface reconstruction without a pre-specified size or a fixed-resolution discretization of the point cloud model. Like GIRA, Voxblox provides CPU[4] and GPU[5] implementations as well as a method to

[2] https://github.com/OrebroUniversity/perception_oru
[3] https://github.com/Octomap/octomap
[4] https://github.com/ethz-asl/voxblox
[5] https://github.com/nvidia-isaac/nvblox

localize within the representation using submaps [155].

Duberg and Jensfelt [57] propose UFOMap, which improves upon Octomap by providing an explicit representation of unknown space and introduces Morton codes for faster tree traversal. An open-source CPU implementation of UFOMap[6] is available; however, the implementation does not provide functionality to localize within the map, like GIRA, Voxgraph, or NDTMap.

Vespa et al. [190] introduce the Supereight mapping framework, which consists of two dense mapping methods a TSDF-based implicit map and an explicit spatial occupancy map. In follow-on work [68], which leverages multi-resolution grids, the authors demonstrate that the TSDF-based method yields superior reconstruction compared to UFOMap. Supereight enables frame-to-model point cloud registration via Iterative-Closest-Point (ICP) alignment. However, Supereight uses RAM to assess memory efficiency, but does not provide statistics on space required to store the representation to disk. The CPU implementation is available for Supereight[7] as open-source software.

Reijgwart et al. [156] have recently proposed the Wavemap hierarchical volumetric representation, which uses wavelet compression for higher memory savings compared to Voxblox, Supereight, and Octomap. Like other discrete mapping methods, the highest resolution of the hierarchical map is set by the user and fixed during robot operation. In contrast, the SOGMM method in GIRA provides the ability to adapt the fidelity of the model according to the complexity of the scene, without utilizing a hierarchical approach [76]. The CPU implementation for Wavemap is available as open-source software[8] and includes some additional improvements (e.g., the use of OpenVDB [131] instead of the octree data structure used in the original paper). The approach, however, does not provide a method to estimate pose.

Doherty et al. [53] propose BGKOctomap, an extension to Octomap, which utilizes nonparametric Bayesian kernel inference for continuous-space occupancy modeling. The method is improved by modeling sensor rays as continuous free-space observations [54]. The CPU[9] implementations of these variants are available as open-source software.

Eckart et al. [61] present compact modeling of point clouds using GMMs to estimate pose. The approach is extended to a hierarchical formulation to improve memory-efficiency and accuracy [60]. Dhawale and Michael [50] present an alternative hierarchical approach, which equally weights the Guassian distributions, for high-resolution surface mapping. Srivastava and Michael [167] present another hierarchical approach but utilize the Expectation-Maximization algorithm to assign non-uniform weights depending on the local density of point cloud data. Common to these approaches is the need to set model complexity criteria such as image patch size [50], component splitting threshold [60], and model fidelity threshold [167]. Further, to the best of our knowledge, there are no open-source software implementations of existing GMM-based point cloud modeling methods. In contrast, we provide open-source CPU

[6] https://github.com/UnknownFreeOccupied/ufomap
[7] https://bitbucket.org/smartroboticslab/supereight2
[8] https://github.com/ethz-asl/wavemap
[9] https://github.com/RobustFieldAutonomyLab/la3dm

and GPU implementations of our work in [76], which leverages information-theoretic learning to adjust the model complexity. Finally, to demonstrate the occupancy modeling and pose estimation capabilities using GMM-based models, we provide CPU implementations based on [178, 176, 180].

A.2 GIRA Design

We envision the GIRA framework to be used in robotics research where 3D perception tasks must be executed in real-time and algorithms for these tasks should be easy to prototype.

Popular robotics software packages like Bullet [44], Drake [182], and TensorFlow [1] provide low-level programming language support for high-performance, real-time operation and high-level programming language bindings for ease of prototyping. We follow the same model with GIRA where key algorithms are implemented in C/C++/CUDA with Python bindings. For C/C++, we use the C/C++17 standard. For GPU support, CUDA version 10.4 and above is required.

To enable message passing between different software systems, most robotics applications use the Robot Operating System (ROS) [150] and its successor ROS2 [117]. The GIRA framework is structured using Collective Construction (`colcon`) packages to help the robotics community easily integrate GIRA within their ROS and ROS2 workspaces. For low-level code and bindings, GIRA utilizes CMake for compilation support on both Linux and macOS. Python virtual environments are used to isolate executables.

For 3D perception tasks, visualization is an important capability for debugging research code. GIRA provides interfaces to the Open3D [211] visualization tools for this purpose. Furthermore, developers can leverage tools like RViz2 from ROS2 after integrating `colcon` packages from GIRA.

A.3 GIRA Framework

The GIRA framework consists of three components: (1) GIRA Reconstruction, (2) GIRA Registration, and (3) GIRA Occupancy Modeling. This section provides an overview of these three components.

A.3.1 GIRA Reconstruction

Given time-synchronized depth and intensity images with known pose estimates, GIRA Reconstruction creates a Self-Organizing Gaussian Mixture Model (SOGMM) [76] that is:

1. **Continuous**, the point cloud is represented with a 4D GMM which is a linear combination of continuous functions (Gaussian distributions).

(a) Input (b) Resampled (c) Intensity Inference

Figure A.2: An example workflow for GIRA Reconstruction Sect. A.3.1. The input is a depth-intensity point cloud shown in (a). The resulting model can be resampled to generate novel 4D points (b) or be used to infer expected intensity values at known 3D locations (c).

2. **Probabilistic**, the 4D GMM captures the variance and expected values for point locations and intensity values.
3. **Generative**, the 4D GMM enables fast sampling of point locations and intensity values from the model.
4. **Compact**, since the number of parameters required to represent the 4D GMM is much lower compared to the point cloud itself.
5. **Adaptive**, the number of Gaussian distributions within the 4D GMM are automatically estimated from the scene complexity of the underlying sensor data.

GIRA Reconstruction utilizes Open3D [211] for point cloud loading, writing, and visualization. We use NumPy [83] for interfacing with Eigen [78] via Pybind11 [91].

GIRA Reconstruction contains CPU and GPU implementations for SOGMM[10]. Both implementations can be accessed via a Python interface:

```
from sogmm_py.sogmm import SOGMM

# SOGMM of pointcloud on CPU
sg_cpu = SOGMM(bandwidth=0.015, compute='CPU')
mcpu = sg_cpu.fit(pointcloud)

# SOGMM of pointcloud on GPU
sg_gpu = SOGMM(bandwidth=0.015, compute='GPU')
mgpu = sg_gpu.fit(pointcloud)
```

where, `pointcloud` is a NumPy array and `bandwidth` is the bandwidth of the kernel used for the Gaussian Blurring Mean Shift (GBMS) within the SOGMM algorithm [76].

These models are continuous and generative. Three-dimensional points along with intensity values can be sampled from the model using:

[10]Detailed tutorials are available at https://gira3d.github.io/docs/index.html.

Platform ID	CPU	GPU	Memory (CPU/GPU)
Ryzen/RTX3090	3960X, 48 threads	RTX 3090	252 GB / 24 GB
Intel/RTX3060	i9-10900K, 20 threads	RTX 3060	32 GB / 12 GB
ARM-12c/Orin	rev 1 (v8l), 12 threads	Orin	32 GB
ARM-8c/Xavier	rev 0 (v8l), 8 threads	Xavier	16 GB
ARM-6c/TX2	Cortex-A57, 6 threads	TX2	8 GB

(a) Target Platforms

```
# Sample 640*480 points from the model
rp = sg_gpu.joint_dist_sample(640*480)
```

A plot of the resampled point cloud is shown in Fig. A.2b.

If the 3D point locations are known, the expected intensity values and variance can be inferred from the model at these locations:

```
# locs is a N x 3 numpy array
# E is N x 1 expected intensities
# V is N x 1 variance
_, E, V = mgpu.color_conditional(locs)
```

A plot of intensity values `E` is shown in Fig. A.2c.

The SOGMM model is compact and its size can be computed as follows.

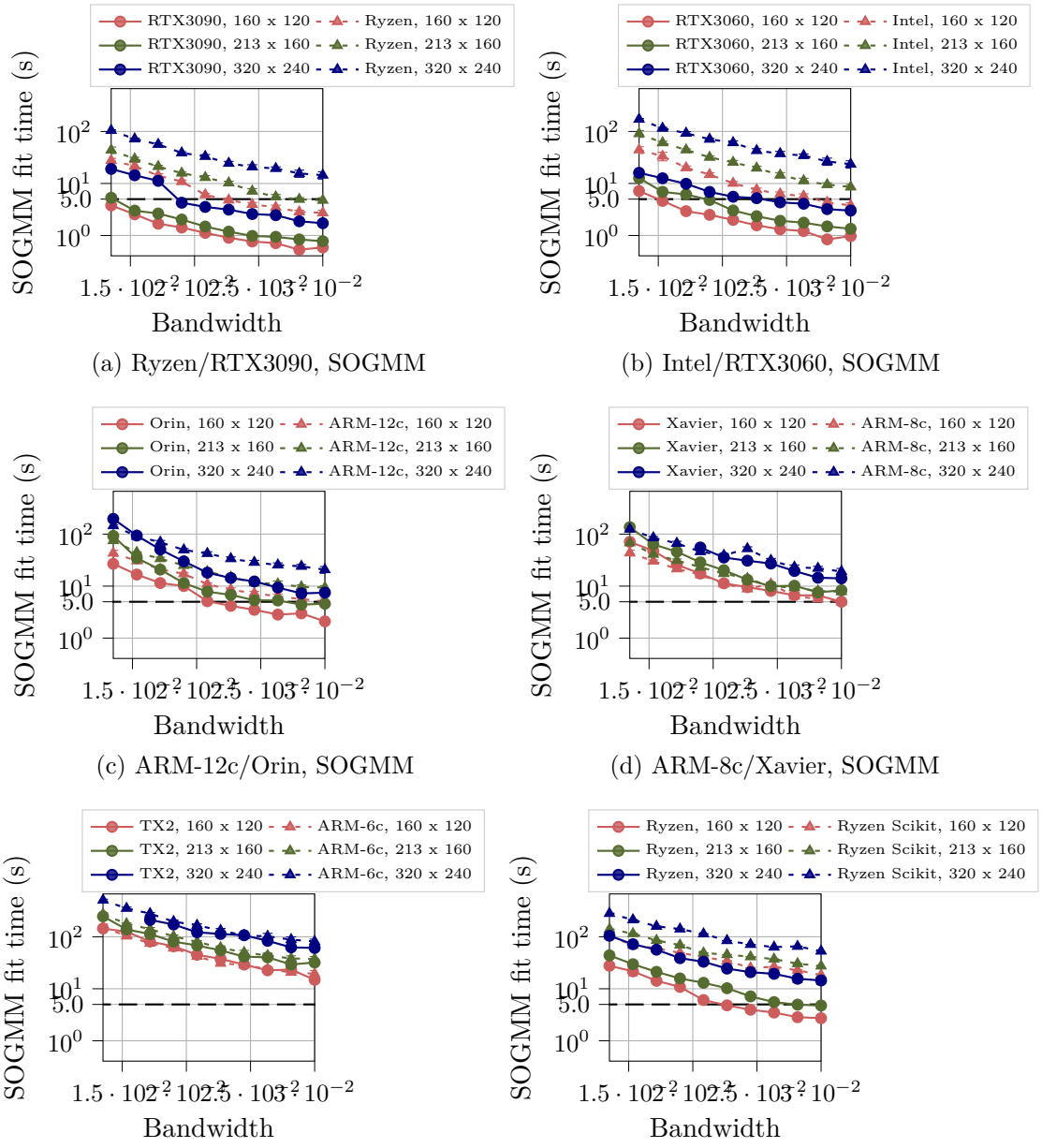
```
# computing memory usage
M = mgpu.n_components_

# 4 bytes per float
# 1 float value per weight
# 4 float values per mean
# 10 float values per covariance
mem_bytes = 4 * M * (1 + 10 + 4)
```

which is 69.78 kilobytes for the model learnt in Fig. A.2.

The time taken to learn a SOGMM is reported as a function of bandwidth parameter for a diverse set of platforms outlined in Fig. A.3a. The input data for this experiment corresponds to frame 854, which was randomly selected, of the simulated `livingroom1` data from the Augmented ICL-NUIM datasets [36]. Ten equally spaced bandwidth values from 0.0135 to 0.0300 are used. Image sizes of 320×240, 213×160, and 160×120 are used (corresponding to $2\times$, $3\times$, and $4\times$ reduction along each axis of the original 640×480 image). Because there is randomness in the KInit step, each case is run ten times and averaged to obtain accurate timing results.

Figures A.3a–A.3e plot the results of these trials for the CPU-only (dashed lines with triangle markers) and GPU-accelerated (solid lines with circle markers) implementations. The y-axes of these plots use a base-10 log-scale and the observed standard deviation, which is plotted as error bars, is very low compared to the mean values. From Figs. A.3a and A.3b we observe over an order of magnitude faster performance when using the GPU-accelerated version of the system for all image sizes. Further, there is an overall decrease in performance from Ryzen/RTX3090 (Fig. A.3a)



(a) Ryzen/RTX3090, SOGMM

(b) Intel/RTX3060, SOGMM

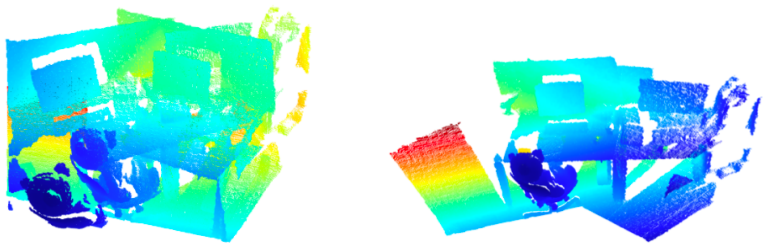
(c) ARM-12c/Orin, SOGMM

(d) ARM-8c/Xavier, SOGMM

(e) ARM-6c/TX2, SOGMM

(f) Comparison with `scikit-learn` on Ryzen

Figure A.3: Comparison of SOGMM computation time via GIRA Reconstruction on the target platforms listed in Fig. A.3a. In (a) and (b) the GPU-accelerated case on the desktop platforms provides more than an order of magnitude improvement in timing compared to the CPU-only case for most image sizes. The results of the embedded platforms shown in (c), (d) and (e) demonstrate that the relative performance improvements seem to degrade with increasing SWaP constraints. In any case, (f) shows that our CPU implementation performs nearly an order of magnitude faster than a reference SOGMM implementation using `scikit-learn`.



(a) Before registration (b) After registration

Figure A.4: The point clouds in (a) are originally misaligned. (b) The code in Sect. A.3.2 estimates the SE(3) transform to align them.

to Intel/RTX3060 (Fig. A.3b) for both CPU-only and GPU-accelerated versions. This is expected due to the decrease in computational capability for both the CPU and GPU.

Figure A.3c provides results for ARM-12c/Orin platform. In this case, the gains for the GPU-accelerated version are lower than the desktop platforms. Notice that for image size 320×240 the CPU starts performing better than GPU at low bandwidths. At low bandwidths, the number of estimated components are high.

Figures A.3d and A.3e suggest a further decrease in relative performance improvement in using the GPU-accelerated version as opposed to the CPU-only version of our system. Further, due to memory constraints the 320×240 image size fails for both platforms below certain bandwidths. Both ARM-8c/Xavier and ARM-6c/TX2 are SWaP-constrained platforms used on robots. For real-world usage of our framework, we recommend using the CPU-only version when CPU resources are not required by other subsystems (e.g., planning, control, and visual-inertial odometry) and using the GPU-accelerated version when CPU resources are in demand (which is often the case).

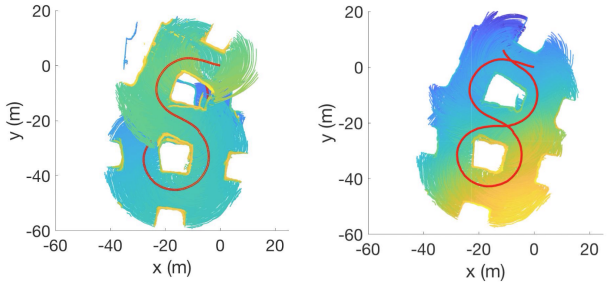
A.3.2 GIRA Registration

This module implements (1) registering a pair of GMMs [178] and (2) closing the loop using a pose graph optimization [176].

The *anisotropic*, *isoplanar*, and *isoplanar-hybrid* registration variants from [178] are implemented in this module. Python and MATLAB interfaces have been developed, but this document provide examples only for the Python interface. The isoplanar-hybrid registration approach first calls a coarse optimizing using the isoplanar registration function followed by a refinement optimization using the anisotropic registration. The source and target variables are paths to files containing GMMs.

```
from gmm_d2d_registration_py import isoplanar_registration
from gmm_d2d_registration_py import anisotropic_registration

# Initial registration guess
```



(a) Without loop closure (b) Loop closure

Figure A.5: The trajectories reconstructed using (a) frame-to-frame registration and (b) with loop closure is enabled are shown with the pointclouds plotted.

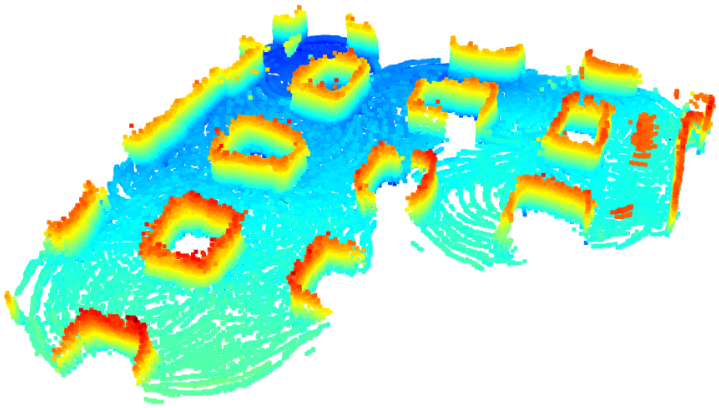


Figure A.6: Resampled points from a GMM are added to an occupancy grid map and the occupied voxels are queried and visualized.

```
Tinit = np.eye(4)

# Isoplanar registration
Tiso = isoplanar_registration(Tinit, source, target)
Tout = anisotropic_registration(Tiso, source, target)

# Rotation and translation solutions
R = Tout[0:3, 0:3]
t = Tout[0:3, 3]
```

The result of registering a single pair of images may be seen in Fig. A.4. In addition, a pose graph optimization example is provided, which uses GTSAM [48]. A comparison of the frame-to-frame registration with and without loop closure is shown in Fig. A.5.

A.3.3 GIRA Occupancy Modeling

This module implements occupancy reconstruction by sampling from a GMM and raytracing through an occupancy grid map. MATLAB and Python interfaces are provided, but only the Python interface is discussed in this document[11]. Like the registration module detailed in Sect. A.3.2, this module is compatible with scikit-learn [144] GMMs and assumes GMMs are loaded from file.

```
# Create 3d occupancy grid with parameters p
grid = Grid3D(p)

# Nx3 sampled from GMM (assumed in world frame)
pts = gmm.sample(num_pts)

# Add the points to the grid
for i in range(0, num_pts):
  ray_end = Point(pts[i,0], pts[i,1], pts[i,2])

  # sensor_pose is in world frame
  # TRIMMED_MAX_RANGE set by user
  grid.add_ray(sensor_pose, ray_end, TRIMMED_MAX_RANGE)
```

Functions for querying occupied, free, and unknown voxels are provided through Python and MATLAB bindings of C++ code. The result of adding sampled points from the Mine dataset GMMs and querying the occupied voxels is shown in Fig. A.6.

A.4 Implementation Details

This section details the GPU-accelerated software architecture of GIRA Reconstruction. Figure A.7 provides an overview of the accelerated SOGMM components [76].

Gaussian Blurring Mean Shift Comaniciu and Meer [39] leverage a binned estimator to determine seeds for the algorithm. A kd-tree [16] is used to query the neighbors in $\mathcal{Y}$. The points within the specified radius are averaged and the seed is updated to the new location. The algorithm terminates when either the number of maximum iterations is reached or there is no substantial change with respect to the previous seed position.

Expectation Maximization The EM algorithm consists of the Expectation (E) and Maximization (M) steps. The E Step evaluates the responsibilities γ_{nb} using the current parameters $\boldsymbol{\mu}_b$, $\boldsymbol{\Sigma}_b$ and π_b via

$$\gamma_{nb} = \frac{\pi_b \mathcal{N}\mathbf{x}_n \boldsymbol{\mu}_b \boldsymbol{\Sigma}_b}{\sum_{a=1}^{M} \pi_a \mathcal{N}\mathbf{x}_n \boldsymbol{\mu}_a \boldsymbol{\Sigma}_a}. \tag{A.1}$$

[11] Detailed documentation for both MATLAB and Python is provided at https://gira3d.github.io/docs/index.html.

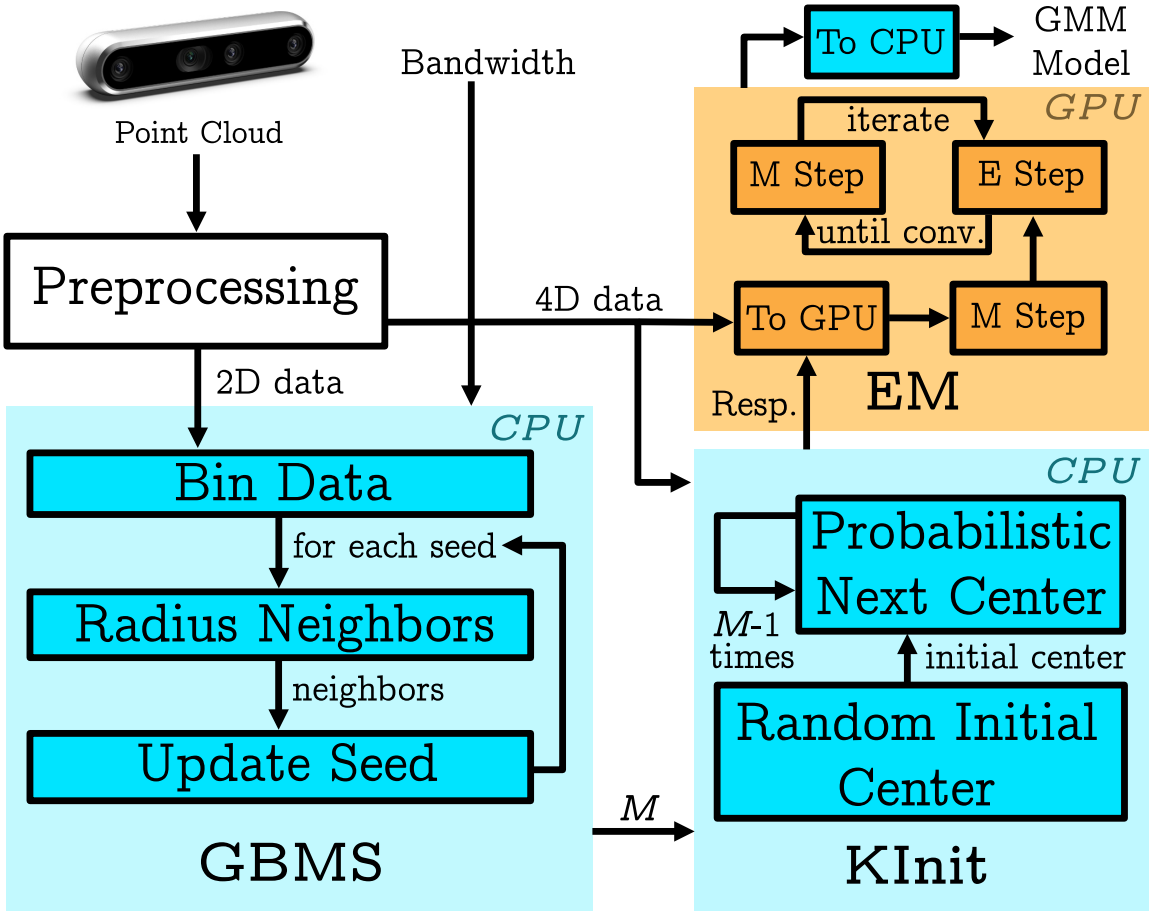


Figure A.7: Information flow for the GPU-accelerated adaptive point cloud modeling system. Given a bandwidth parameter and depth-intensity image pair, the Gaussian Blurring Mean Shift (GBMS) obtains the number of components M. The number of components and the 4D data are used by KInit to calculate the responsibility matrix used by the EM algorithm. The result of the EM algorithm is the SOGMM model [76].

To reduce the computational complexity of Eq. (A.1), the natural logarithm can be applied to convert the multiplications and divisions into sums and differences:

$$\begin{aligned} \ln \gamma_{nb} = \ln \pi_b + \ln \left(\mathcal{N} \mathbf{x}_n \boldsymbol{\mu}_b \boldsymbol{\Sigma}_b \right) \\ - \ln \left(\sum_{a=1}^{M} \pi_a \mathcal{N} \mathbf{x}_n \boldsymbol{\mu}_a \boldsymbol{\Sigma}_a \right). \end{aligned} \tag{A.2}$$

Term 2 of Eq. (A.2) may be rewritten, as derived in [23, 145, 175],

$$\begin{aligned} &\ln \left(\mathcal{N} \mathbf{x}_n \boldsymbol{\mu}_b \boldsymbol{\Sigma}_b \right) \\ &= -\frac{1}{2} \left(D \ln(2\pi) + \sum_{j=1}^{D} \left(\mathrm{P}_b (\mathbf{x}_n - \boldsymbol{\mu}_b) \right)_j^2 \right) + \left(\sum_{j=1}^{D} \ln \left(\mathrm{diag}(\mathrm{P}_b) \right)_j \right) \end{aligned} \tag{A.3}$$

where $\boldsymbol{\Sigma} = \mathrm{L}\mathrm{L}^\top$, $\mathrm{P} = \mathrm{L}^{-1}$, and L is a lower triangular matrix calculated using the Cholesky decomposition of the covariance matrix. Summing the logarithm of the

diagonal entries of P_b (i.e., $\ln(\mathrm{diag}(\mathrm{P}_b))$) in Eq. (A.3) is equivalent to $\ln |\boldsymbol{\Sigma}_b|^{-1/2}$.

The GPU implementation leverages higher-order tensor representations (rank-3 and rank-4 tensors)[12]. The weights are represented as a rank-3 tensor of shape $(1, M, 1)$, means are represented as a rank-3 tensor of shape $(1, M, 4)$, and covariances are represented as a rank-4 tensor of shape $(1, M, 4, 4)$. This implementation accelerates unary (e.g., logarithm and exponential of a matrix, reduction operations like summing along a dimension or taking a maximum along a dimension of a rank-2 or a rank-3 tensor) and binary (e.g., addition, subtraction, multiplication, and division of rank-2 and rank-3 tensors) operations via element-wise CUDA kernels with fixed block and grid sizes for all GPUs.

Rank-2 tensor multiplication is accelerated using the `cuBLAS`[13] `gemm` routine. Rank-3 tensor multiplication is accelerated via the `cuBLAS` `gemmStridedBatched` routine. The Cholesky decomposition of a rank-2 tensor is accelerated via the `cuSOLVER`[14] `potrf` routine. The Cholesky decomposition of a rank-3 tensor is accelerated using the `cuSOLVER` `potrfBatched` routine. Using the Cholesky decomposition, a linear system of equations involving rank-2 tensors is solved using the `cuSOLVER` `potrs` routine and for rank-3 tensors using the `potrsBatched` routine.

A.5 Summary

GIRA is a set of tools and software for processing point cloud data into Gaussian mixture models for inference and robot autonomy. These tools and software are released open-source https://github.com/gira3d. Fundamental robotics capabilities from our prior works on point cloud modeling [76], pose estimation [178, 176], and occupancy modeling [180] are included in the open-source release. These fundamental capabilities have applications beyond reconstruction and aerial robotics. The adaptivity of the SOGMM representation has applicability to perception in the small and fine grained manipulation tasks. The variable resolution occupancy grid mapping and distribution to distribution registration software may be leveraged for high-speed mobile robot applications like off-road operations. By releasing this software, the authors hope to increase the accessibility of these formulations to technical experts.

[12] For the exposition of the GPU-accelerated components, tensor conventions from TensorFlow [1] are used.

[13] `cuBLAS` https://docs.nvidia.com/cuda/cublas

[14] `cuSOLVER` https://docs.nvidia.com/cuda/cusolver