UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Novel Non-feature Based SLAM: Joint Optimization of Non-feature Maps and Robot Trajectories

by Yingyu Wang

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of
Prof. Shoudong Huang
A/Prof. Liang Zhao

University of Technology Sydney
Faculty of Engineering and Information Technology

July 2025

Certificate of Original Authorship

I, Yingyu Wang, declare that this thesis is submitted in fulfillment of the requirements for the award of Doctor of Philosophy, in the School of Mechanical and Mechatronic Engineering, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship doi.org/10.82133/C42F-K220.

Signature: Production Note: Signature removed prior to publication.

Date: 30 July 2025

Novel Non-feature Based SLAM: Joint Optimization of Non-feature Maps and Robot Trajectories

by

Yingyu Wang

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

In unknown environments, robots estimate their own pose while simultaneously building a map of their surrounding environment using onboard sensors. This problem, known as simultaneous localization and mapping (SLAM), is a fundamental task in robotics that has been extensively studied over the past several decades. Recently, with growing computational resources and increasing demand for spatial understanding in autonomous systems, non-feature-based SLAM has gained significant attention. Compared to feature-based approaches, non-feature map representations typically provide denser environmental information and are better suited for supporting downstream robotic tasks. For example, occupancy grid maps (OGMs) encode obstacle presence probabilities at each cell, while signed distance functions (SDFs) represent distances to the nearest surface, both providing rich spatial context.

Most existing non-feature-based SLAM methods decouple localization and mapping: robot poses are optimized independently from map construction. Typically, pose graph optimization is employed to reduce cumulative errors, relying on relative pose estimates from front-end systems. While computationally efficient, this separation limits the consistency between the estimated poses and the resulting maps. Joint optimization of poses and features in feature-based SLAM has shown substantial improvements in both accuracy and consistency. However, joint optimization strategies for non-feature-based SLAM remain

underexplored. This thesis addresses this gap by investigating joint optimization frameworks that simultaneously optimize both robot poses and non-feature maps, with a focus on OGMs and SDFs.

This work presents three main contributions:

First, we formulate OGM-based SLAM as a joint optimization problem that simultaneously estimates robot poses and occupancy values. To the best of our knowledge, this is the first work that jointly optimizes robot pose and position-independent non-feature map representations within a unified optimization framework. We parameterize the occupancy probabilities at observed locations and include them, alongside the robot poses, in a nonlinear least squares (NLLS) problem. A variant Gauss-Newton (GN) solver is proposed for this formulation, which is implemented in both 2D and 3D cases. To improve convergence and robustness in 2D, a multi-resolution strategy is introduced, using coarser-to-finer occupancy maps across iterations. Experimental results on both simulated and real-world datasets demonstrate that our approach outperforms state-of-the-art methods in accuracy and achieves improved efficiency in 2D.

While effective on normal-scale datasets, the first method's computational complexity grows with trajectory length. As the robot travels further, the dimensionality of the optimization problem increases, making it more difficult to solve. To address this scalability issue, the second contribution introduces an OGM-based submap joining framework. Here, the global OGM and the poses of local submaps are jointly optimized. The objective is formulated as a NLLS problem minimizing discrepancies in obstacle probabilities between the global map and aligned local submaps. The number of times each cell is observed is used as a confidence-weighted parameter to guide the optimization process. A key theoretical insight is that in the GN iterations, pose updates are independent of global occupancy values. Based on this, we derive a pose-only GN solver that is provably equivalent to the full GN method but significantly more efficient. Once optimal poses are estimated, the global map can be reconstructed in closed form. Experiments in 2D confirm the method's superior accuracy and efficiency, and 3D experiments validate its accuracy.

Having established the value of joint optimization of poses and OGM, the third contribution explores its applicability to 3D SDF-based SLAM. SDFs provide distance fields,

making them highly suitable for tasks like planning and reconstruction. We propose a joint optimization framework that integrates robot poses and a 3D truncated SDF (TSDF) map. Unlike previous approaches that rely on point-to-SDF or SDF-to-SDF registration and optimize only robot poses, our method jointly optimizes both the robot poses and the entire TSDF map. Specifically, it incorporates three types of constraints—surface, space, and coplanar—formulated based on the properties of the TSDF field. Critically, we avoid explicit plane extraction or data association by implicitly embedding coplanar relationships within the TSDF structure and dynamically updating them during optimization. This enables robust, adaptive optimization without requiring manual feature extraction. Experiments on real-world 3D datasets confirm the method's superior accuracy and robustness.

In conclusion, the three contributions of this thesis advance the field of non-feature-based SLAM by introducing and validating joint optimization frameworks for OGMs and SDFs. Extensive experiments on simulated and real-world datasets show that these methods yield significant improvements in accuracy compared to traditional decoupled non-feature-based SLAM approaches. The proposed frameworks offer a promising new direction for enhancing localization and mapping in non-feature-based SLAM by tightly coupling pose and non-feature map estimation.

Acknowledgements

Time flies, and yet the past four years at the University of Technology Sydney (UTS) have been among the most rewarding and transformative experiences of my life. As I reflect on this journey, I am filled with deep gratitude for the friendships, support, guidance, and inspiration that have accompanied me throughout. Pursuing a Ph.D. at UTS has been an incredibly meaningful chapter that I will always treasure.

First and foremost, I would like to express my heartfelt appreciation to my supervisors, Professor Shoudong Huang and Associate Professor Liang Zhao. I am truly grateful for the opportunity they gave me to undertake my Ph.D. and for their consistent support, insightful guidance, and generous mentorship throughout my research journey. Their academic rigor, clarity of thought, and integrity have had a profound influence on me. During times of uncertainty, their encouragement and belief in my potential gave me the confidence to persevere. Their passion for research continues to inspire me and will guide my path forward.

I am also sincerely thankful for the collegial and welcoming environment at the UTS Robotics Institute. I would like to especially thank my friends and labmates, Tiancheng Li, Yang Song, Mengya Xu, Qiao Wang, Yiran Zhou, Yang Xu, and Ziqi Wang, for their companionship, support, and countless moments of encouragement. The collaborative and relaxed atmosphere of our lab made the Ph.D. journey not only productive but also enjoyable and memorable. These friendships are among the most valuable outcomes of my time at UTS.

Finally, I would like to express my deepest gratitude to my family for their unconditional love and unwavering support. Their constant encouragement has given me strength, and their faith in me has been a steady source of motivation. I could not have made this journey without them by my side.

To everyone who has supported me along the way, thank you.

Contents

List of Figures

List of Tables

Acronyms & Abbreviations

SLAM	Simultaneous Localization and Mapping
OGM	Occupancy Grid Map
SDF	Signed Distance Function
NeRF	Neural Radiance Field
PGO	Pose Graph Optimization
NLLS	Non-linear Least Squares
GN	Gauss-Newton
2D	Two-dimensional
3D	Three-dimensional
GP	Gaussian Process
ESDF	Euclidean Signed Distance Function
TSDF	Truncated Signed Distance Function
NDT	Normal Distributions Transform
ICP	Iterative Closest Point
BA	Bundle Adjustment
MAE	Mean Absolute Error

RMSE Root Mean Squared Error

AUC Area Under the ROC Curve

DoF Degree-of-Freedom

ATE Absolute Trajectory Error

APE Absolute Pose Error

Nomenclature

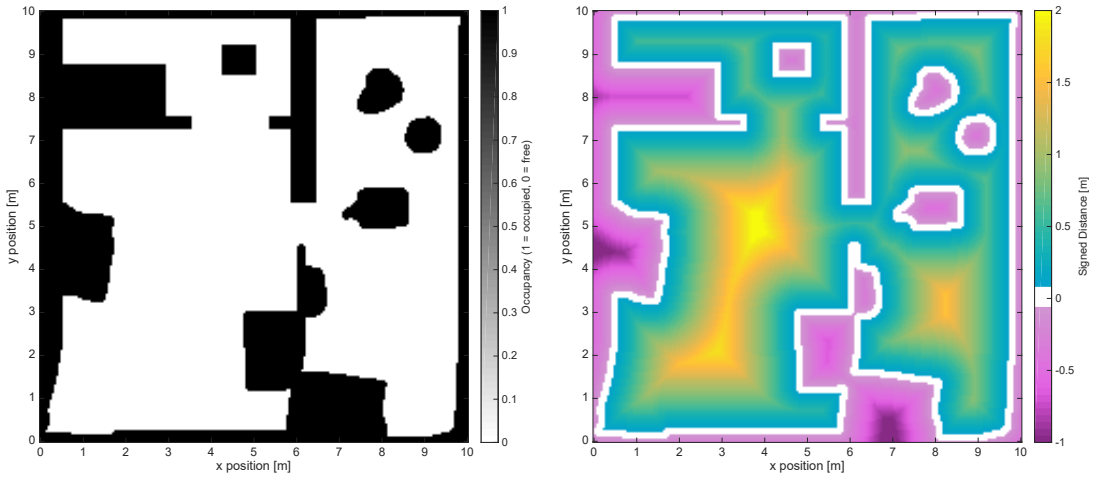
	General Notations
$\mathbb{M}$	Set of occupancy values at discrete cell vertices in an OGM map
$\mathbb{O}$	Set of all odometry inputs
$\mathbb{S}_i$	Set of sensor observations at timestep i in OGM-based approach
$\mathbb{N}$	Set of hit numbers at all discrete cell vertices in an OGM map
$\mathbb{Z}$	Set of sensor observations in SDF-based approach
$M(\cdot)$	Function that queries the occupancy value at an arbitrary position in an OGM
$N(\cdot)$	Function that queries the hit number at an arbitrary position via linear interpolation from the hit map $\mathbb{N}$
$D(\cdot)$	Function that queries a signed distance value at an arbitrary position in a SDF map
$\mathbf{x}$	State vector for optimization
$\mathbf{x}^P$	Vector of all robot poses to be optimized
$\mathbf{x}^M$	Vector of occupancy values for all cell vertices in the discrete map $\mathbb{M}$
$\mathbf{x}^D$	Vector of signed distance values for cell vertices in a TSDF map
$\mathbf{m}$	Discrete coordinates of a cell vertex
$\mathbf{p}$	Continuous coordinates of a point
$\mathbf{n}$	Normal vector
ξ	Rotation vector
$(\cdot)^\wedge$	Skew-symmetric matrix corresponding to the Lie algebra vector

Chapter 1

Introduction

In unknown environments, robots perceive their surroundings and construct maps while simultaneously estimating their own poses within these maps, a problem known as simultaneous localization and mapping (SLAM) [1]. As a fundamental task in robotics, SLAM has been extensively studied over the past several decades, as it not only provides robots with the critical ability of localization and environment perception but also facilitates various downstream tasks such as navigation, path planning, exploration, and scene understanding. SLAM systems typically adopt diverse types of map representations. Among these, geometric feature-based representations, such as point features, line features, and plane features, are among the most commonly utilized. Due to the availability of mature techniques for feature extraction and data association, these feature-based approaches have been widely implemented in SLAM frameworks, e.g., MonoSLAM [2], ORB-SLAM [3], PL-SLAM [4], and LOAM [5]. However, feature-based SLAM methods often exhibit limitations in environments characterized by low texture or repetitive structural patterns, leading to reduced robustness and accuracy. Additionally, feature-based maps tend to produce sparse representations and inherently lack the capability to densely encode environments, thereby posing difficulties for certain downstream robotic tasks.

Non-feature map representations have attracted increasing attention in recent years, driven

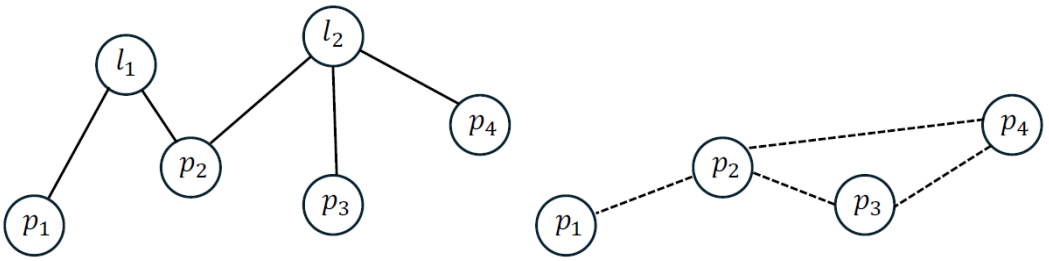


(a) OGM (b) SDF Map

FIGURE 1.1: **Example of 2D OGM and SDF** [6]. In the OGM, colors indicate occupancy values ranging from 0 (free) to 1 (occupied). In the SDF, colors represent distances to the nearest obstacle surface: white corresponds to the observed object surface, green and yellow denote free space outside the object, and pink indicates regions inside the object.

by advances in computational power and the growing demand for comprehensive environmental understanding in robotics. Unlike feature-based maps, which represent environments through sparse geometric parameters extracted from distinct features, non-feature maps employ representations that do not explicitly rely on feature extraction. For example, as illustrated in Figure 1.1(a) and Figure 1.1(b), occupancy grid maps (OGMs) discretize space into grids and assign each cell a probability of occupied, while signed distance function (SDF) maps encode the environment as a continuous field representing the signed distance to the nearest surface. Other notable non-feature representations include neural radiance fields (NeRFs) and Gaussian splatting, both of which leverage dense and continuous parameterizations to model complex scenes with high fidelity.

In general, non-feature maps are typically denser and encode richer geometric and semantic information than feature maps, making them particularly valuable for downstream robotic tasks. To capitalize on these advantages, non-feature-based SLAM approaches use such map representations to improve both mapping quality and localization accuracy. These representations have been leveraged to enhance various SLAM components, including odometry estimation [7], loop closure detection and optimization [8, 9], and submap



(a) Joint Optimization of Poses and Features (b) Pose Graph Optimization

FIGURE 1.2: **Example of joint optimization of poses and features and pose graph optimization in feature-based SLAM.** Solid lines represent observations, and dashed lines represent relative pose transformation estimated from front-end techniques.

joining [10, 11], ultimately enabling more accurate and robust localization. Given these benefits, SLAM research built on non-feature map representations has received growing attention and popularity in recent years.

1.1 Motivation

As a robot moves, errors in odometry estimation inevitably accumulate, leading to trajectory drift. To address this issue, batch optimization techniques are widely applied in feature-based SLAM systems. Two common strategies are the joint optimization of poses and features (Figure 1.2(a)) and pose graph optimization (PGO, Figure 1.2(b)). The concept of joint optimization refers to estimating robot poses and map variables simultaneously within a single optimization problem, analogous to bundle adjustment (BA) in visual SLAM [12, 13]. In contrast, PGO simplifies the problem by replacing pose–feature constraints with relative pose constraints, typically obtained from front-end modules such as odometry estimation or loop closure detection, and optimizes only the poses. This makes PGO a computationally efficient approximation of the joint optimization, albeit with some loss of accuracy. In general, the joint optimization provides more accurate and consistent estimates of both poses and features. Consequently, many feature-based SLAM systems employ a combination of PGO and the joint optimization, often at different stages or scales (e.g., local and global), to balance efficiency with accuracy.

In non-feature based SLAM, PGO is also widely adopted to reduce accumulated error. While some approaches still leverage feature-based front-end methods to extract relative pose constraints to formulate PGO problems and then construct non-feature maps, a growing number of methods directly utilize non-feature maps for relative pose estimation and then formulate PGO. For example, pose constraints can be derived from OGM-based scan-to-map matching [14], OGM-based loop closure detection [8], and SDF-based submap registration [10]. However, a key limitation of these approaches is that the uncertainty in pose estimation is not accounted for during the mapping process, potentially resulting in inconsistent and inaccurate pose and map estimates.

In contrast to joint optimization in feature-based SLAM, research on the joint optimization of robot poses and non-feature maps remains limited. This is largely due to the fundamental differences in representation: in feature-based SLAM, both observations and maps are parameters of features, such as positions, making their relationship well-defined and tractable for optimization. In contrast, for non-feature representations, the relationship between observations and maps is more complex, often specific to the particular type of map (e.g., OGMs and SDF maps). This complexity makes it difficult to formulate a uniform joint optimization framework, as exemplified in Figure 1.2(a). There are some other challenges in the joint optimization of poses and non-feature maps, including the lack of well-established data association techniques, the influence of map resolution on optimization accuracy and efficiency, and the computational complexity of solving large-scale problems.

Given the gap and challenges in non-feature based SLAM, the goal of this thesis is to investigate how to jointly optimize robot poses and non-feature maps in SLAM framework, with a focus on approaches based on OGMs and SDF maps.

1.2 Main Contributions

The main contributions of this thesis are summarized below:

i. ***Chapter 3***:

- We formulate the OGM-based SLAM problem as a joint optimization problem where the poses and the occupancy map are optimized together.
- We propose a variation of Gauss-Newton (GN) method to solve the proposed formulation, enabling the estimation of more accurate robot poses and occupancy maps compared to existing state-of-the-art techniques.
- To enhance efficiency, convergence, and robustness in two-dimensional (2D) SLAM, we propose a multi-resolution optimization strategy using occupancy maps of different resolutions across stages.
- In the 2D case, our method achieves robust convergence even with keyframes of limited overlap, outperforming state-of-the-art approaches like Cartographer in efficiency while maintaining superior accuracy.
- We extend our method to three-dimensional (3D), with preliminary results demonstrating superior accuracy compared to other state-of-the-art approaches.

ii. ***Chapter 4***:

- We formulate the OGM-based submap joining problem as a NLLS problem where both local submap coordinate frames and the global occupancy map are considered as state variables to be optimized simultaneously.
- We prove that, when solving the formulated NLLS for both poses and the global map using GN method, the increment of poses in each iteration is independent of the occupancy values of the global occupancy map.
- Based on the independent property, we propose a pose-only GN method which is equivalent to full GN method for solving the proposed formulation. The optimal global occupancy map can be obtained using a closed-form formula after the optimal poses are obtained. Experiments demonstrate that the pose-only GN algorithm is very efficient and much faster than the full GN method.
- In the 2D case, experiments on both simulated and real-world datasets demonstrate that the proposed method outperforms existing state-of-the-art approaches in terms of both efficiency and accuracy.
- The proposed method is further extended to the 3D case, with preliminary results showing improved accuracy over existing state-of-the-art approaches.

iii. ***Chapter 5***:

- We formulate the local SDF-based BA problem as a NLLS problem, in which both the robot poses and the TSDF map are treated as state variables and jointly optimized.
- The proposed method leverages the fundamental properties of the TSDF field to define surface, space, and coplanar constraints within the TSDF space. This allows the optimization to utilize the full volumetric information of the TSDF map, rather than focusing solely on its zero-level set. This represents a fundamental difference from previous SDF-based SLAM approaches.
- The coplanar constraint term implicitly encodes coplanar relationships within the TSDF properties. These constraints are dynamically evaluated and updated during the optimization process, eliminating the need for explicit data association prior to optimization and improving robustness and adaptability.
- Experimental results demonstrate that the proposed method outperforms state-of-the-art LiDAR odometry and planar-feature-based BA techniques in both robustness and accuracy. Moreover, the proposed approach is significantly more efficient than our previous proposed OGM-based joint optimization method, while achieving comparable accuracy, highlighting the advantages of jointly optimizing robot poses with the TSDF map.

1.3 Thesis Organization

This thesis is organized as follows:

- **Chapter 2**: In this chapter, some significant contributions in areas of non-feature map representations, non-feature based SLAM, submap joining, and joint optimization of poses and maps are briefly reviewed.
- **Chapter 3**: In this chapter, 2D and 3D OGM-based SLAM problems are formulated as joint optimization problems where the poses and the occupancy map are optimized together. A variation of GN method is proposed to solve the proposed formulation,

enabling the estimation of more accurate robot poses and occupancy maps. This work is based on our Robotics: Science and Systems (RSS), 2022 paper [15], and includes selected content from our IEEE Transactions on Robotics (T-RO), 2025 [16] paper.

- **Chapter 4**: In this chapter, 2D and 3D OGM-based submap joining problems are formulated as a NLLS problem where both local submap coordinate frames and the global occupancy map are considered as state variables to be optimized simultaneously. We prove a special independence property of the formulation and, based on this property, propose a pose-only GN method that is mathematically equivalent to the full GN method but significantly more efficient. This work is based on our IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024 paper [17], and includes selected content from our IEEE Transactions on Robotics (T-RO), 2025 paper [16].

- **Chapter 5**: In this chapter, the joint optimization of robot poses and 3D TSDF maps is investigated. The proposed framework leverages the fundamental properties of the SDF field to define surface, space, and coplanar constraints within the TSDF space. Unlike conventional SDF-based methods that perform point-to-SDF or SDF-to-SDF registration, the proposed approach directly integrates geometric and field-level constraints from the TSDF and jointly optimizes both poses and whole TSDF space, enabling more comprehensive utilization of spatial information. Moreover, unlike traditional plane feature-based BA approaches that require explicit planar feature extraction and manual data association, this proposed method embeds coplanar relationships implicitly within the TSDF structure. These coplanar constraints are dynamically evaluated and updated during optimization, enabling adaptive and robust data association without relying on explicit features.

- **Chapter 6**: In this chapter, the conclusions of all chapters are summarized and presented. In addition, we present the potential future research directions of joint optimization of poses and non-feature maps.

1.4 List of Publications

1.4.1 Related Publications

[1] **Yingyu Wang**, Liang Zhao, and Shoudong Huang, "Occupancy-SLAM: An Efficient and Robust Algorithm for Simultaneously Optimizing Robot Poses and Occupancy Map", in *IEEE Transactions on Robotics (T-RO)*, vol. 41, pp. 4057-4077, 2025, doi: 10.1109/TRO.2025.3578227.

[2] **Yingyu Wang**, Liang Zhao, and Shoudong Huang, "Grid-based Submap Joining: An Efficient Algorithm for Simultaneously Optimizing Global Occupancy Map and Local Submap Frames", in 2024 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates, 2024, pp. 10121-10128, doi: 10.1109/IROS58592.2024.10802536.

[3] Liang Zhao, **Yingyu Wang**, and Shoudong Huang, "Occupancy-SLAM: Simultaneously Optimizing Robot Poses and Continuous Occupancy Map", in *Proceedings of Robotics: Science and Systems (RSS)*, New York City, NY, USA, 2022, doi: 10.15607/RSS.2022.XVIII.003.

1.4.2 Other Publication

[1] Yiran Zhou, **Yingyu Wang**, Shoudong Huang, and Liang Zhao, "Correspondence-Free Multiview Point Cloud Registration via Depth-Guided Joint Optimisation", Accepted by *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Chapter 2

Preliminaries and Related Works

A fundamental requirement for robots operating in unknown environments is the ability to construct internal representations of their surroundings, as well as to estimate their own poses. These capabilities serve as the foundation for a wide range of downstream tasks, including obstacle avoidance, path planning, and high-level decision-making. Such map representations encode the presence and spatial configuration of obstacles, objects, and other relevant entities relative to the robot's poses. This problem is known as SLAM, in which the poses and the map are inherently coupled and should remain mutually consistent. To improve computational efficiency, many SLAM systems employ front-end methods, such as data association and loop closure detection, to approximate pose-to-map relationships using pose-to-pose constraints, thereby reducing the problem to a simplified PGO. While this strategy significantly lowers computational cost, it may compromise the consistency between the estimated poses and the map due to uncertainties introduced by the front-end modules.

SLAM methods can be broadly categorized into two groups based on whether geometric features are explicitly extracted from the underlying map representation: feature-based SLAM and non-feature-based SLAM. In feature-based SLAM, the map is represented by parameters of geometric features (e.g., their positions). In contrast, non-feature based SLAM does not rely on explicit feature extraction, and the map is represented using alternative forms such as occupancy values or signed distance fields.

To provide the necessary background and avoid potential confusion, this chapter outlines key concepts and methods related to different map representations, non-feature based SLAM approaches, and the distinction between PGO and joint optimization of poses and maps. In addition, related work on submap joining is also presented.

2.1 Non-feature Representations

Early map representations in SLAM focused on the extraction of a few salient features from the environment, such as point, line, and plane features. However, as computational power increases and the need for spatial information grows, non-featured map representations are receiving increasing attention because they are critical for downstream tasks that require detailed knowledge of the environment (e.g., planning, navigation, and decision-making).

Non-feature map representations are spatial representations that do not rely on explicitly extracted geometric features to describe the environment, and whose parameterized forms are therefore independent of such features. Instead, they model the environment as dense grids or continuous fields, typically encoding information over the entire spatial domain. Common examples include OGMs which represent space as a probabilistic grid of occupied or free regions, and SDFs which encode the distance from each spatial location to the nearest surface. Unlike feature-based representations, non-feature maps retain richer geometric or semantic details of the environment.

In this section, we introduce non-feature map representations such as the OGM and the SDF map, along with their respective mapping techniques. We also introduce other representative non-feature map formats and conclude with a comparative analysis of OGMs and SDFs.

2.1.1 OGM and Occupancy Mapping

OGM first proposed by [18–20] is one of the widely used non-feature map representations in robotic tasks for its ability to clearly represent obstacles, free space, and unknown areas, facilitating collision-free navigation and path planning. The classical OGM divides the

environment into a set of discrete grid cells, each of which stores the probability that the corresponding location in the environment is occupied by an obstacle. This probability is typically computed by propagating the uncertainty of sensor observations using Bayes' rule. We first introduce the classical Bayesian-based occupancy update model and occupancy mapping method, and then summarize the developments and variants of OGMs.

2.1.1.1 Classical Bayesian-based Occupancy Update Model

We present the classical Bayesian-based occupancy update method [21] in 2D as an example.

Occupancy mapping typically begins by assuming that the robot poses are known, addressing the problem of generating consistent maps from noisy and uncertain sensor measurements. Classical Bayesian occupancy mapping is essentially a grid-based mapping technique, where the environment is discretized into several grid cells. The fundamental idea of OGMs is to represent the map as a field of binary random variables, each corresponding to the occupancy state of its respective grid cell. Occupancy grid mapping algorithms estimate the posterior probability distribution of these random variables based on sensor data.

Given sensor measurements $\mathbb{S}_{1:n}$ up to timestamp n and corresponding sensor poses $\mathbb{X}^P_{1:n}$, the occupancy state of a grid cell $\mathbf{m}$ is modeled as a Bernoulli random variable. We denote the event that cell $\mathbf{m}$ is occupied as $\text{occ}(\mathbf{m})$. The posterior occupancy probability is then written as $p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n})$. Applying Bayes' rule yields

$$p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n},) = \frac{p(\mathbb{S}_n \mid \text{occ}(\mathbf{m}), \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n}) p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n})}{p(\mathbb{S}_n \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n})}. \tag{2.1}$$

Under the standard Markov assumption [21], measurements are conditionally independent given the map and the current pose:

$$p(\mathbb{S}_n \mid \text{occ}(\mathbf{m}), \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n}) = p(\mathbb{S}_n \mid \text{occ}(\mathbf{m}), \mathbb{X}^P_n). \tag{2.2}$$

Thus, the posterior becomes

$$p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n}) = \frac{p(\mathbb{S}_n \mid \text{occ}(\mathbf{m}), \mathbb{X}^P_n) p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1})}{p(\mathbb{S}_n \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n})}. \tag{2.3}$$

Applying Bayes' rule again results in

$$p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n}) = \frac{p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n) p(\mathbb{S}_n \mid \mathbb{X}^P_n) p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1})}{p(\text{occ}(\mathbf{m})) p(\mathbb{S}_n \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n})}. \tag{2.4}$$

Similarly, the free-space probability is

$$p(\text{free}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n}) = \frac{p(\text{free}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n) p(\mathbb{S}_n \mid \mathbb{X}^P_n) p(\text{free}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1})}{p(\text{free}(\mathbf{m})) p(\mathbb{S}_n \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n})}. \tag{2.5}$$

Taking the ratio and canceling common terms yields

$$\begin{aligned}
&\frac{p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n})}{p(\text{free}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n})} \\
&= \frac{p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n) p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}) p(\text{free}(\mathbf{m}))}{p(\text{free}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n) p(\text{free}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}) p(\text{occ}(\mathbf{m}))} \\
&= \frac{p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n\right)}{1 - p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n\right)} \frac{p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}\right)}{1 - p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}\right)} \frac{1 - p(\text{occ}(\mathbf{m}))}{p(\text{occ}(\mathbf{m}))}
\end{aligned} \tag{2.6}$$

To simplify computation, we take the natural logarithm of both sides as

$$\begin{aligned}
&ln \frac{p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n})}{1 - p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n})} \\
&= ln \frac{p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n\right)}{1 - p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}^P_n\right)} + ln \frac{p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}\right)}{1 - p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}\right)} + ln \frac{1 - p(\text{occ}(\mathbf{m}))}{p(\text{occ}(\mathbf{m}))},
\end{aligned} \tag{2.7}$$

where $ln \frac{1-p(\text{occ}(\mathbf{m}))}{p(\text{occ}(\mathbf{m}))}$ is prior, and $ln \frac{p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}\right)}{1-p\left(\text{occ}(\mathbf{m}) \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}\right)}$ is the previous estimate. With the common assumption of a uniform prior $p(\text{occ}(\mathbf{m})) = 0.5$, the prior term vanishes.

Letting

$$\mathcal{L}(\mathbf{m} \mid \cdot) = \log \frac{p(\text{occ}(\mathbf{m}) \mid \cdot)}{1 - p(\text{occ}(\mathbf{m}) \mid \cdot)}, \tag{2.8}$$

the recursive update (2.7) simplifies to the classical log-odds form:

$$\mathcal{L}(\mathbf{m} \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n}) = \mathcal{L}(\mathbf{m} \mid \mathbb{S}_{1:n-1}, \mathbb{X}^P_{1:n-1}) + \mathcal{L}(\mathbf{m} \mid \mathbb{S}_n, \mathbb{X}^P_n). \tag{2.9}$$

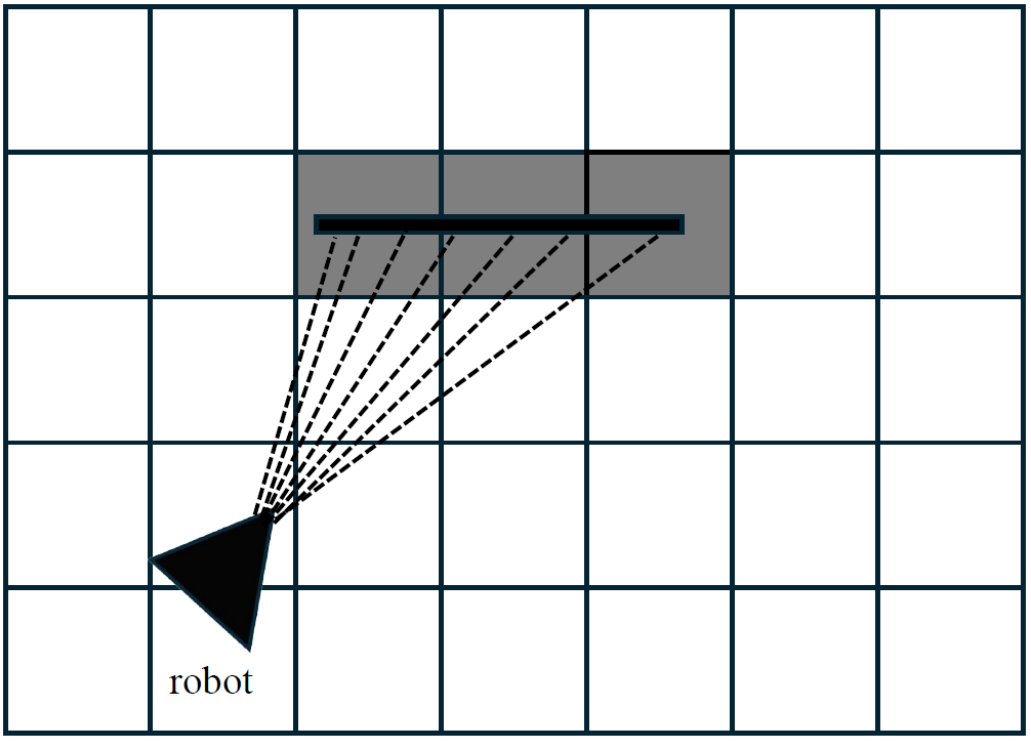


FIGURE 2.1: **An example of the ray casting algorithm in the 2D case.** Dotted lines represent the beams of a 2D laser scanner, black rectangles indicate obstacles. Gray cells denote occupied space, while white cells represent free space.

Note that occupancy values in log-odds form can be converted to probabilities and vice versa. Therefore, the log-odds value is typically stored for each cell instead of the probability itself. Assuming that the occupancy probabilities of all cells are independent, modeling the occupancy values for all individual cells is equivalent to modeling the entire OGM.

In (2.9), the current estimate of an observed cell is updated based on sensor measurements. Specifically, if a cell is traversed by a ray from the sensor origin to the hit point (endpoint), it is considered to be in a free state; otherwise, it is regarded as occupied. Various methods exist for modeling sensor observations in occupancy mapping, among which ray-casting is the most widely adopted, as illustrated in Figure 2.1.

Depending on the choice of inverse sensor model, the probability $p(\text{occ}(\mathbf{m}) \mid \mathbb{S}_n, \mathbb{X}_n^P)$ takes different values based on whether the cell is observed as occupied or free. A commonly used model for LiDAR sensors is the beam-based inverse sensor model, which assumes that measurement endpoints correspond to obstacle surfaces and that the space between

the sensor origin and the endpoint is free of obstacles. Based on this model, the occupancy probability is typically set to 0.7 for occupied cells and 0.4 for free cells [21, 22].

After computing occupancy values using the log-odds formulation in (2.9), a threshold is usually applied to convert the continuous occupancy value $\mathcal{L}(\mathbf{m} \mid \mathbb{S}_{1:n}, \mathbb{X}^P_{1:n})$ into a binary state. That is, a cell is classified as occupied if the occupancy value exceeds the threshold, and as free otherwise, thereby discretizing the environment into two distinct states.

2.1.1.2 Variants of OGMs and Occupancy Mapping

OGMs, first proposed in 2D by [18–20], offer a probabilistic representation of the environment by classifying space into free, occupied, and unknown regions. Due to their ability to explicitly model sensor uncertainty and spatial state, OGMs have become a foundational representation in a wide range of robotic applications, leading to the development of various extensions and variants.

With the increasing demand for 3D mapping in robotics, OGMs have been extended to 3D space. However, this extension significantly increases memory consumption due to the dense nature of 3D grids. To address this challenge, multi-resolution strategies are introduced in OGM representation and occupancy mapping techniques. Payeur et al. [23] introduce the use of octrees for probabilistic 3D occupancy mapping, enabling a more compact representation of free and occupied space. Similar strategies are adopted by [24] and [25], though these works did not explicitly address map compression or bounded confidence in the stored probabilities.

The most widely used 3D occupancy mapping framework today is OctoMap [22], which employs a memory-efficient hierarchical octree structure to represent 3D OGMs. To further improve mapping scalability and adaptability, approaches such as RMAP [26] and ColMap [27] incorporate adaptive-resolution techniques that dynamically refine grid resolution based on local complexity. More recently, Reijgwart et al. [28] propose a wavelet-based hierarchical compression scheme for occupancy maps, enabling efficient storage, incremental updates, and fast querying.

Another major trend in occupancy mapping is the shift from discrete occupancy grid representations to continuous occupancy representations. Unlike standard occupancy grid maps, which discretize the environment into independent grid cells, continuous occupancy maps model the occupancy of space as a continuous distribution and explicitly capture the correlations among spatial regions. Such continuous formulations preserve richer structural information and better represent environmental uncertainty, thereby improving the performance of robotic tasks such as exploration and navigation.

Various methods have been proposed to realize continuous occupancy representations. For instance, Gaussian Mixture Models are employed by [29] to represent environmental structures continuously. MRFMap [30] adopts a Markov Random Field to integrate depth information, enabling real-time construction of continuous 3D occupancy maps. Gaussian Process (GP)-based methods utilize the mean and variance surfaces of GP to continuously represent uncertainty. The original GP-based occupancy mapping formulation, introduced in [31, 32], follows a batch-processing framework and was later extended to incremental formulations in [33, 34]. However, online GP training suffers from poor scalability, as its computational complexity grows cubically with the number of data points. To address this limitation, several Bayesian and low-rank formulations have been proposed. Hilbert Map [35] reformulates continuous occupancy mapping as a binary classification problem in a reproducing kernel Hilbert space. This approximation yields an implicit low-rank representation that enables scalable training on large datasets. The Sequential Bayesian Hilbert Map (SBHM) [36] extends this idea by applying a Bayesian treatment that incrementally updates the map from sequential observations. SBHM further achieves sparsity by computing feature vectors based on a sparse set of hinge points, typically arranged on a coarse grid. To enhance scalability in multi-robot scenarios, Zhi et al. [37] propose a decentralized framework that merges Bayesian Hilbert Maps constructed by multiple robots. Their method introduces an efficient approximation, termed Fast Bayesian Hilbert Maps (Fast-BHM), which reduces the computational complexity to below quadratic, thereby improving mapping efficiency while preserving mapping accuracy. The Sparse Bayesian Kernel-based Mapping (SBKM) framework [38] extends kernel-based occupancy mapping by introducing a sparse Bayesian formulation that enables efficient and probabilistic modeling of continuous environments. The method employs an incremental Relevance Vector

Machine learning algorithm to update the map online from streaming observations, with occupancy probabilities modeled via a probit likelihood function.

With the advent of deep neural networks, several learning-based approaches have been proposed for occupancy mapping. Occupancy Networks [39] learns a continuous 3D occupancy function through a neural implicit representation, enabling high-resolution and memory-efficient reconstruction with superior mesh quality compared to voxel-based methods. Building on this idea, subsequent works extend neural implicit representations to dynamic scene reconstruction [40] and occupancy prediction in autonomous driving scenarios [41].

2.1.2 SDFs and SDF Mapping

2.1.2.1 SDFs

Another widely used non-feature map is the SDF, which represents the environment as a grid where each cell stores the distance to the nearest surface. The surface of an object is implicitly defined by the zero-crossings of the distance values [42]. Positive values indicate space in front of the surface, negative values represent space behind it, and the surface itself lies at the transition where the sign changes. A 2D SDF is demonstrated in Figure 2.2 [43].

Depending on the task requirements, various forms of SDFs are utilized. Euclidean signed distance fields (ESDFs), which store the Euclidean distance from each voxel to the nearest obstacle, are widely adopted in optimization-based robotics applications such as path planning [44], manipulation [45], and registration [46, 47] due to their ability to provide high-quality proximity gradients. To improve computational efficiency, truncated signed distance fields (TSDFs) have become popular in robotics and computer vision applications [42, 48].

Unlike ESDFs, which rely on true Euclidean distances, TSDFs encode projective distances, i.e., the distance along the sensor's line of sight to the observed surface. These distances are computed only within a narrow truncation band around the surface, significantly

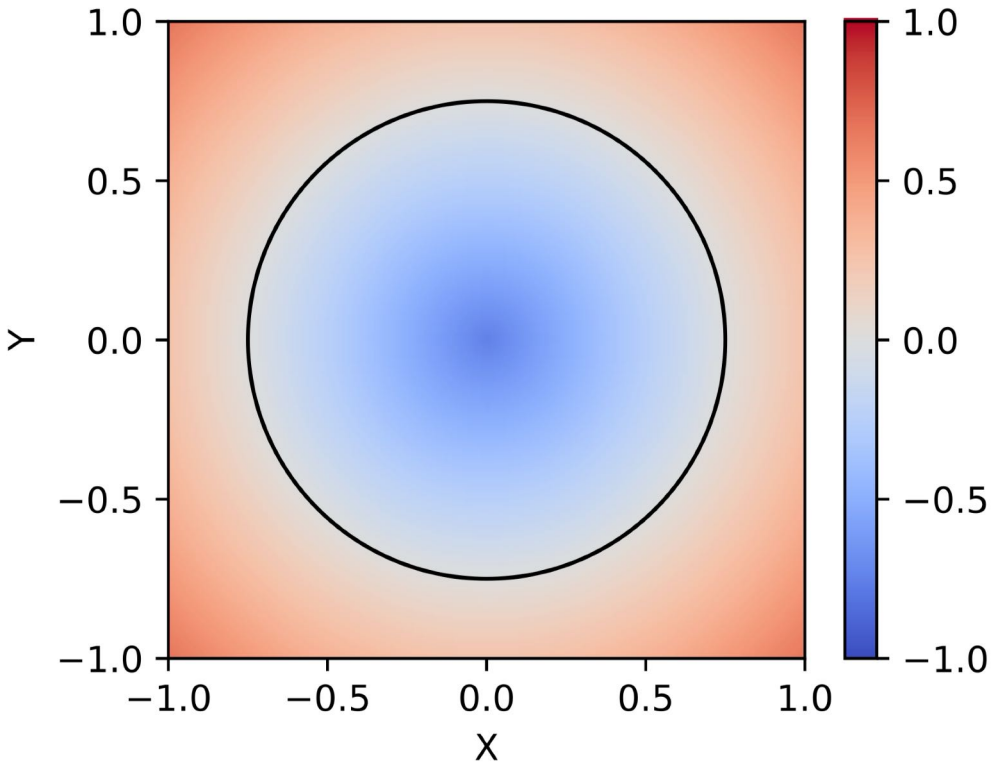


FIGURE 2.2: **An example of 2D SDFs [43].** The SDF function takes negative values inside the object (blue) and positive outside (red). The zero-crossing of the function corresponds to the object's surface.

reducing the computational load. This truncated representation is more suitable for 3D reconstruction, where only the zero-level set of the distance field is needed to extract the surface geometry to meshes using methods such as Marching Cubes [49].

2.1.2.2 SDF Mapping

According to the definition of SDFs introduced in Section 2.1.2.1, when all observations and their corresponding poses are available, constructing SDFs is straightforward: one can compute the Euclidean distance from each cell to its nearest surface and assign a sign based on whether the cell lies inside or outside the object, this can efficiently be done using techniques such as fast marching [50], fast sweeping method [51] and the more general level-set method [52–54]. However, in the context of SLAM, where the robot incrementally explores the environment and only partial observations are available, building SDF maps

becomes considerably more challenging. An incremental mapping approach is therefore required to continuously update the signed distance values as new sensor data arrives.

The most widely adopted incremental SDF mapping approach is the KinectFusion paradigm [48]. This method integrates each new scan into a TSDF by ray-casting from the sensor origin to every point in the input scan and updating the distance and weight estimates along each ray. The choice of weighting function plays a critical role in the accuracy of the resulting reconstruction, particularly in the case of large cells, where a single cell may be updated by thousands of points from a single scan.

Specifically, at timestamp i, the existing signed distance and weight values for a cell at location $\mathbf{m}$ are denoted as $D_{i-1}(\mathbf{m})$ and $W_{i-1}(\mathbf{m})$, respectively. Given an incoming 3D point $\mathbf{q}_{ij}$ and the position of sensor $\mathbf{t}_i$, the updated signed distance $D_i(\mathbf{m})$ and weight $W_i(\mathbf{m})$ are computed as

$$\begin{aligned} D_i(\mathbf{m}) &= \frac{W_{i-1}(\mathbf{m}) D_{i-1}(\mathbf{m}) + w_{ij}^{mq} d_{ij}^{mq}}{W_i(\mathbf{m}) + w_{ij}^{mq}}, \\ W_i(\mathbf{m}) &= \min\left(W_i(\mathbf{m}) + w_{ij}^{mq}, w_{\max}\right), \end{aligned} \tag{2.10}$$

where w_{ij}^{mq} is the weight of the current measurement, $w_{\max}$ is the hyperparameter which means the upper bound of the total weight, and

$$d_{ij}^{mq} = \|\mathbf{q}_{ij} - \mathbf{m}\| \operatorname{sign}((\mathbf{q}_{ij} - \mathbf{m}) \cdot (\mathbf{q}_{ij} - \mathbf{t}_i)). \tag{2.11}$$

$\operatorname{sign}(\cdot)$ denotes the sign function, which returns a positive value if the input is positive, negative if the input is negative, and zero otherwise.

KinectFusion originally suggests using weights w_{ij}^{mq} that depend on the angle between the ray from the sensor origin and the estimated surface normal. This angular weighting aims to downweight observations with oblique incidence, which are typically less reliable. However, KinectFusion notes that for high-resolution volumetric maps with millimeter-scale voxels, a constant weight of 1 is sufficient to produce accurate reconstructions. In contrast, Voxblox [44] introduces a more refined weighting scheme to improve reconstruction quality. Their method aims to reduce the contribution of voxels that lie behind the observed

surface and are thus not directly measured. This strategy mitigates the effect of uncertain updates and enhances the fidelity of the resulting surface geometry.

While this incremental SDF mapping approach enables efficient construction of TSDFs, it is limited to representing only the truncated region around observed surfaces and does not provide full Euclidean distance information throughout the environment. As a result, it is unsuitable for tasks that require global distance information, such as exploration and path planning. To address this limitation, Lau et al. [55] propose an efficient method for incrementally constructing ESDFs from binary OGMs. Building on this idea, Voxblox [44] extends the approach by enabling the direct construction of ESDFs from TSDFs.

In addition to the aforementioned SDF mapping approaches, a number of recent works further extend signed distance field representations for robotic perception and planning. FIESTA [56] proposes a fast incremental ESDF framework tailored for online motion planning of aerial robots, significantly improving update efficiency in large-scale environments. More advanced continuous distance-field formulations include Log-Gaussian Process implicit surfaces for generating faithful Euclidean distance fields [57], as well as sparse Gaussian process based incremental metric-semantic mapping for multi-robot systems [58]. Neural implicit methods also gain traction: iSDF [59] demonstrates real-time neural SDF mapping for robot perception, and more recent systems such as MISO [60] leverage multi-resolution submaps to achieve globally consistent neural implicit reconstruction. These works collectively highlight the broader landscape of SDF mapping, spanning analytic, probabilistic, and neural representations.

2.1.3 Comparisons Between OGMs and SDFs

As two commonly used non-feature map representations in SLAM, both OGMs and SDFs estimate spatial quantities throughout the observed cells. Despite this shared characteristic, they emphasize fundamentally different aspects of environmental modeling, and the choice between them largely depends on the intended application [43]. There are two key distinctions between the two non-feature map representations.

Directness of Modeling: OGMs are well-suited to modeling spatial states directly from sensor measurements. Since each measurement ray provides explicit information about free, occupied, or unobserved space, occupancy probabilities can be updated with relatively few assumptions. In contrast, SDFs focus on modeling the signed distance to a surface. While the distance can be computed precisely when the surface is fully known, it must be approximated under partial observations, typically using heuristics like the TSDF.

Smoothness and Differentiability: SDFs inherently produce smooth representations, as they encode continuous-valued functions rather than discrete occupancy states. This smoothness offers several advantages: it enables differentiability, allowing the computation of useful spatial gradients; it supports interpolation, yielding sub-cell resolution; and it mitigates discretization artifacts. However, the inability to represent discontinuities sharply means that SDFs may fail to capture fine structures or thin obstacles present in the environment.

2.1.4 Other Non-feature Map Representations

Beyond OGMs and SDFs, a variety of other non-feature map representations have been proposed, each with distinct properties suited to specific applications. Notable examples include surfels [61], meshes, Normal Distributions Transform (NDT) maps [62], Neural Radiance Fields (NeRF) [63], and Gaussian Splatting [64]. These representations differ fundamentally in how they encode spatial and appearance information and thus serve different purposes in robotic perception and reconstruction.

For instance, mesh-based representations are well-suited for explicit object reconstruction and surface modeling. NDT and OGM approaches are more appropriate for long-term mapping tasks, especially in large-scale environments. SDFs provide continuous and intuitive distance information, which is valuable for motion planning and optimization. In contrast, neural representations such as NeRF and Gaussian Splatting excel in high-fidelity rendering and scene appearance modeling, making them particularly useful in vision-centric applications.

2.2 Non-feature based SLAM

In Section 2.1, we introduced several representative non-feature map representations and their corresponding mapping techniques. In this section, we demonstrate how these non-feature representations, when used in SLAM, can not only enhance the mapping process but also improve localization accuracy.

2.2.1 OGM-based SLAM

One of the most widely adopted non-feature based SLAM approaches is OGM-based SLAM, where OGMs serve as the representation of the environment. Early and classical methods in this category were predominantly developed for 2D LiDAR-based applications. Notable examples include FastSLAM [65] and GMapping [66], both of which leverage particle filters to perform simultaneous localization and mapping. These approaches excel at handling data association in cluttered environments but often suffer from high computational overhead and scalability issues, particularly in large-scale, long-duration operations. To overcome these limitations, modern optimization-based approaches, such as Hector-SLAM [14], Karto-SLAM [67], and Cartographer [8], are proposed. Hector-SLAM adopts a scan-to-map matching strategy without relying on odometry, offering real-time performance in indoor environments. However, its lack of loop closure mechanisms limits its applicability to small-scale settings. Karto-SLAM incorporates loop closure detection with sparse pose adjustment for global optimization. Cartographer further advances this line of research by integrating real-time scan matching with a PGO framework. It employs a branch-and-bound technique for efficient loop closure detection and correction, enabling robust performance in large-scale, real-world environments.

In these 2D OGM-based SLAM approaches, OGMs serve not only as the primary representation for building the environment but also as a means to improve localization accuracy. These maps support a range of downstream robotic tasks such as path planning and obstacle avoidance. For localization, methods like Hector-SLAM, Karto-SLAM, and Cartographer utilize scan-to-map matching to estimate accurate odometry, thereby reducing

cumulative pose errors. Additionally, loop closure detection in Karto-SLAM and Cartographer relies on the spatial structure and probabilistic nature of OGMs to recognize previously visited locations and correct global pose estimates.

These examples demonstrate that using OGMs as the map representation in SLAM can enhance performance compared to feature-based approaches. A key advantage is that they avoid the need for explicit data association, both in odometry estimation and loop closure detection, which simplifies implementation and increases robustness in ambiguous or feature-sparse environments.

Beyond 2D applications, occupancy grid representations have been integrated with 3D SLAM in order to satisfy the requirements of perception, exploration, and path planning, particularly in LiDAR and RGB-D based SLAM. Yet most existing systems employ 3D OGMs only as a mapping process. A common pipeline first estimates odometry with a LiDAR odometry module and then integrates the resulting point clouds into OctoMap [22] or similar 3D occupancy mapping techniques. The map itself is not fed back to improve pose estimation. For example, RTAB-Map [68] can generate 3D OGMs for navigation, but the map is isolated for mapping rather than being tightly coupled to the localization process.

2.2.2 SDF-based SLAM

Another category of non-feature SLAM approaches is SDF-based SLAM. Due to the continuous gradient field and the ability to reconstruct accurate surfaces via Marching Cubes [49], SDF representations are widely used in both visual-based and LiDAR-based SLAM, as well as in various downstream tasks.

Beyond their role in 3D reconstruction, SDFs have been leveraged in various SLAM systems to improve both localization accuracy and mapping quality. In particular, SDFs have proven beneficial for tracking. The seminal work by [42] introduces the idea of camera tracking using SDFs, where camera motion is estimated via the iterative closest point (ICP) algorithm [69] in a coarse-to-fine manner. For each incoming frame, a point cloud is first rendered from the SDF at the previous pose using ray tracing and is then aligned

with the current depth image using projective data association [70] and point-to-plane error metrics. Building upon this idea, Bylow et al. [71] propose a real-time system that avoids explicit data association and downsampling steps typically required by traditional ICP, achieving faster and more efficient performance. EM-Fusion [72] further adopts a similar approach for camera tracking but introduces a robust Huber loss to mitigate the impact of outliers. Additionally, EM-Fusion utilizes the similarity between TSDF surfaces for probabilistic data association, enabling the system to handle dynamic objects effectively. Gradient-SDF [73] integrates SDFs with normals of voxel in a surfel-like representation and demonstrates that this combination benefits both camera tracking and photometric BA in SLAM. By incorporating geometric and gradient information, Gradient-SDF enhances pose estimation accuracy and provides more robust constraints during global optimization, leading to improved overall SLAM performance. InfiniTAM v1 and v2 [74, 75] similarly leverage TSDF representations for real-time SLAM, combining projective ICP tracking with efficient voxel-hashing-based volumetric fusion to achieve scalable dense mapping.

A strong indication that SDF representations can enhance localization rather than merely supporting mapping is the effective use of multi-resolution TSDF structures to improve overall SLAM performance. Supereight [76] presents a unified framework that integrates tracking, mapping, and planning using an octree-based TSDF representation. Camera tracking is performed by aligning incoming frames to the TSDF map using the ICP algorithm. A follow-up study [77] extends this framework by introducing an adaptive-resolution octree structure, which enables finer scene representation and improved noise robustness. These enhancements contribute to higher localization accuracy and improved mapping quality.

2.2.3 Other Non-feature Based SLAM

Other non-feature-based map representations have also been adopted in SLAM systems to address the limitations of feature-based methods, particularly in textureless, dynamic, or geometrically complex environments. These include mesh-based [78–80], NDT based [81–83], NeRF based [84–86] and Gaussian Splatting based [87–89].

Mesh-based SLAM approaches explicitly reconstruct surfaces and object boundaries, offering high-fidelity scene representation well-suited for applications such as robot-object interaction and semantic mapping. NDT-based methods, on the other hand, model the environment using continuous probability distributions in each voxel, facilitating scan registration and enabling robust localization in large-scale or repetitive scenes. NeRF-based SLAM systems leverage implicit volumetric radiance fields to achieve photo-realistic 3D scene reconstruction and enable localization via view synthesis, bridging the gap between geometry and appearance. Recently, Gaussian Splatting has emerged as a lightweight yet expressive alternative to NeRF, supporting real-time rendering and efficient mapping while maintaining visual quality.

Although these approaches vary in representation forms and implementation strategies, they share the common goal of improving environmental modeling, enhancing localization accuracy, or achieving both. Their diversity reflects the growing interest in task-specific map structures that move beyond features, offering new possibilities for robust, dense, and semantically meaningful SLAM systems.

However, all the optimization-based SLAM approaches that utilize non-feature based maps need to optimize the poses first and then build the non-feature based map using the optimized poses. This separation prevents these approaches from jointly considering the uncertainties in both localization and mapping during the optimization process. In contrast, this thesis considers unifying the optimization of both the robot poses and occupancy values at each cell vertex of the occupancy map into a single optimization problem, which can be expected to yield better accuracy.

2.3 Submap Joining

Another closely related topic discussed in this thesis is submap joining, originally proposed by [90]. This technique has become a widely adopted strategy in SLAM, particularly for large-scale environments, owing to its computational efficiency and its ability to mitigate the risk of getting trapped in local minima, which is an issue commonly encountered in full pose graph optimization. By dividing the environment into smaller, locally consistent

submaps and subsequently aligning them, submap joining provides a scalable framework for both mapping and localization. In this section, we review the evolution of submap joining techniques, beginning with early approaches that employ geometric features as map representations. These feature-based methods rely on explicit data association for aligning submaps, which often becomes fragile in textureless or ambiguous environments. We then explore more recent developments in non-feature based submap joining.

Feature-based submap joining approaches have been extensively studied over the years, with many demonstrating both computational efficiency and high estimation accuracy [91–94]. For instance, Huang et al. [91] propose the Sparse Local Submap Joining Filter, which exploits the sparsity of the SLAM information matrix and employs an information filter to efficiently merge submaps. Linear SLAM [92] introduces a method that transforms the inherently non-linear SLAM problem into a sequence of linear least squares problems through submap joining, significantly improving computational tractability. More recently, FS-SLAM [94] employs Fourier series to parameterize closed-shape features in 2D laser SLAM, enabling compact and accurate feature representation, which facilitates more reliable submap alignment and merging.

To extend non-feature-based SLAM to large-scale environments and long-term operations, recent efforts have explored non-feature-based submap joining methods that do not rely on explicit features. For example, Wagner et al. [95] divide the environment into overlapping submaps composed of small TSDF grids, as in KinectFusion [48]. Submap joining is then formulated as a PGO problem, where submap poses are nodes and relative transformations from ICP serve as edges. Similarly, VOG-map [96] represents submaps as 3D occupancy grids, converts them to point clouds for ICP based relative transformations, and solves submap joining via PGO. This approach allows for globally deformable maps that can correct accumulated drift through loop closures while maintaining free space information for path planning. C-blox [97] advances this idea by incrementally generating local TSDF volumes, each with an independent coordinate frame. Its key innovation lies in an entropy-based submap fusion strategy: instead of fusing all overlapping submaps indiscriminately, it estimates pose uncertainty using covariance from the pose graph and fuses only when doing so increases global consistency. This selective approach balances scalability with reconstruction fidelity. InfiniTAM v3 [98] also introduces a submap-based formulation,

in which multiple TSDF volumes are created during dense RGB-D tracking and their relative poses are refined through background PGO. Unlike the ICP-based map-to-map constraints used in earlier TSDF or occupancy submap joining approaches, the inter-submap constraints in InfiniTAM v3 arise directly from the dense tracking pipeline.

The properties of non-feature-based map representations can be directly exploited in submap joining to obtain more accurate relative pose estimates, thereby improving overall performance. Voxgraph [10] improves accuracy by employing SDF-to-SDF registration for overlapping submaps created with Voxblox [44]. By exploiting the underlying SDF representation, Voxgraph generates correspondence-free constraints between submap pairs, enabling efficient optimization. Unlike time-sequence-based submap partitioning, Wang et al. [11] use spatial partitioning, merging submaps during loop closures by solving a pose graph containing only submap frames, with reconstruction decisions based on environmental changes. This elastic approach allows for dynamic adjustment of the map structure in response to changes in the environment, facilitating long-term and large-scale SLAM operations.

All the aforementioned non-feature-based submap joining approaches estimate relative measurements between overlapping submaps to formulate and solve the PGO problem to obtain submap frames. In contrast, this thesis aims to research jointly optimizing submap frames (poses) and non-feature maps.

2.4 Joint Optimization of Poses and Maps

Joint optimization of poses and maps is closely related to the topic of this thesis. Compared to PGO, joint optimization can achieve higher accuracy, as it directly utilizes the information contained within the maps without resorting to approximations that convert the relationships between poses and maps into simplified pose-to-pose relationships, as discussed in Section 1.1. In this section, we summarize and review relevant literature on joint optimization techniques for poses and maps.

The concept of joint optimization of poses and maps is analogous to BA in visual SLAM [12, 13]. In BA, both the camera poses and, optionally, the intrinsic parameters are

optimized alongside the 3D positions of observed points to minimize the reprojection error, that is the difference between the observed image points and the projected points from the estimated 3D structure [99]. In traditional feature-based visual SLAM [100–104], this process involves establishing pairwise correspondences across images and estimating the relative camera motion by minimizing the reprojection error with respect to camera poses and 3D feature positions. This concept has been extended to direct methods, where instead of relying on feature correspondences, a photometric loss function, measuring pixel-wise intensity differences, is directly minimized with respect to both the camera motion and 3D structure. Such approaches are commonly referred to as photometric BA or direct BA [105–107].

Recently, the concept of BA has been extended to LiDAR-based SLAM. However, applying the idea of visual-based BA directly to LiDAR data presents unique challenges. Unlike visual sensors, LiDAR produces sparse point clouds, making feature point extraction more difficult. Moreover, LiDAR sensors lack photometric information, further limiting the applicability of visual SLAM techniques. To overcome these limitations, several approaches have adapted the BA to operate on planar features instead of point features. For example, Kaess [108] and Hsiao et al. [109] minimize the discrepancy between observed and predicted planes, whereas [110–113] focus on minimizing the Euclidean distance between scan points and their corresponding predicted planes. Based on the idea of minimizing Euclidean distance between points in scans, BALM [114] demonstrates that planar parameters can be solved analytically in closed form, reducing the dimensionality of the optimization. BALM2 [115] further improves efficiency by using point clusters, avoiding individual point enumeration. HBA [116] introduces a hierarchical structure to address the scalability challenges of BALM and BALM2 in large environments.

In summary, joint optimization of poses and features (feature-based BA) and joint optimization of poses and points (direct BA) are well-established methods. Features and points naturally associate spatial positions, sensor observations, and poses, thus allowing straightforward integration into optimization frameworks. In contrast, formulating effective constraints that link observations, poses, and non-feature-based map representations (e.g., OGMs and SDFs) for joint optimization remains a significant challenge.

Only a limited number of studies have investigated the joint optimization of poses and non-feature-based maps. One notable effort is Kimera-PGMO [78], which integrates pose optimization with mesh deformation. It constructs a deformation graph from a simplified mesh and combines it with a pose graph, formulating the problem as a factor graph solvable using GTSAM [117]. While Kimera-PGMO shares a similar motivation with this thesis, to improve both map quality and pose accuracy through joint optimization, it fundamentally differs in its choice of map representation. Meshes, composed of vertex positions and their interconnections, naturally support factor graph formulations.

Zhang et al. [118] also addresses joint optimization of camera poses and a mesh-based map, where the mesh is defined by vertex intensities. In their approach, optimization is performed over camera poses and vertex intensities. Although the representation is mesh-based, its formulation closely resembles direct BA, as it lacks explicit geometric constraints between vertices and focuses on photometric consistency.

Another notable work is BAD SLAM [119], which represents the scene with surfels and jointly refines surfel geometry and camera poses using geometric and photometric residuals. Rather than performing a full joint optimization, BAD SLAM adopts an alternating scheme that updates surfels and poses separately, resulting in an approximate form of BA suitable for real-time operation. While highly effective for producing accurate surface reconstructions, this approach models only the visible surfaces without representing free space or volumetric structure, and it largely targets reconstruction quality rather than downstream robotic tasks such as planning or exploration. Moreover, its reliance on good initialization and view overlap may limit robustness in more challenging conditions.

More recently, Zhou et al. [120] propose a multiview point cloud registration method that jointly optimizes camera poses and depth maps. Unlike traditional photometric approaches, their method does not assume photometric constancy. Instead, it directly uses depth constraints to refine a global depth map. This approach shares a similar motivation with this thesis, but depth values are inherently tied to the spatial coordinates of points (e.g., z-axis), making the optimization formulation more straightforward than in occupancy-based or implicit representations.

2.5 Chapter Summary

Joint optimization of poses and maps, commonly referred to as BA in visual SLAM, has been extensively studied over the past few decades. Unlike PGO, which typically relies on approximations that convert constraints between poses and maps into pose-to-pose constraints, joint optimization leverages the available information more directly. As a result, it often produces more accurate solutions. However, most existing research focuses on either feature-based maps or direct methods that optimize over point-based representations. As robotic applications increasingly demand spatial understanding for downstream tasks such as exploration, path planning, and manipulation, the need for representations that capture spatial information, such as occupancy and distance, has grown significantly. This thesis explores the joint optimization of poses and non-feature-based maps, specifically OGMs and SDFs. The goal is to achieve higher accuracy than traditional non-feature-based SLAM approaches, which typically separate the optimization into two stages: first estimating poses and then assuming those poses are correct when building the map.

The primary distinction between the joint optimization of poses and features or points, and that of poses and non-feature-based maps lies in the nature of the relationships between poses and the map representations. In feature-based or direct SLAM, the connection between observations and the map is relatively straightforward, as both are parameterized by spatial positions. In contrast, non-feature-based SLAM requires case-specific formulations, as the relationship depends heavily on the parameterization of each map representation. Moreover, feature-based SLAM benefits from mature front-end techniques for data association. However, in non-feature-based SLAM, especially when jointly optimizing both poses and map representations, data association strategies must be carefully re-examined and tailored to the particular formulation. Finally, the diverse properties inherent in different non-feature-based representations demand corresponding adaptations in optimization methods to ensure both efficiency and robustness. These limitations make the joint optimization of poses and non-feature maps still an open problem for us.

To address the limitations discussed above, this thesis investigates various techniques for the joint optimization of poses and non-feature-based maps. Specifically, we propose joint optimization methods for poses and OGMs in both 2D and 3D cases, aiming to

solve the SLAM problem with higher accuracy. To handle the computational challenges posed by large-scale environments and long-duration trajectories, we further introduce an OGM-based submap joining framework. This method jointly optimizes the submap coordinate frames and the global OGM in both 2D and 3D scenarios. Recognizing the different requirements of downstream tasks, we also develop a hierarchical SDF-based joint optimization framework. This framework unifies the optimization of local poses and local SDFs with the global trajectory and a consistent global SDF map.

The goal of this thesis is to advance SLAM systems that incorporate occupancy or distance-based representations within a tightly coupled optimization framework, where the uncertainties of mapping and pose estimation are jointly considered to mutually improve both components.

Chapter 3

Joint Optimization of Robot Trajectories and Occupancy Maps

Joint optimization of poses and features has been extensively studied and demonstrated to yield more accurate results in feature-based SLAM problems. However, research on jointly optimizing poses and non-feature-based maps remains limited. Occupancy maps are widely used non-feature-based environment representations because they effectively classify spaces into obstacles, free areas, and unknown regions, providing robots with spatial information for various tasks. In this chapter, we propose Occupancy-SLAM, a novel optimization-based SLAM method that enables the joint optimization of robot trajectory and the occupancy map through a parameterized map representation.

The key novelty lies in optimizing both robot poses and occupancy values at different cell vertices simultaneously, a significant departure from existing methods where the robot poses need to be optimized first before the map can be estimated. In our formulation, the state variables in optimization include all the robot poses and the occupancy values at discrete cell vertices in the occupancy map. A variation of GN method is proposed to solve the optimization problem to obtain the optimized occupancy map and robot trajectory.

In this chapter, we first present the formulation of joint optimization of poses and OGMs in the 2D case. We then extend this approach to the 3D scenario.

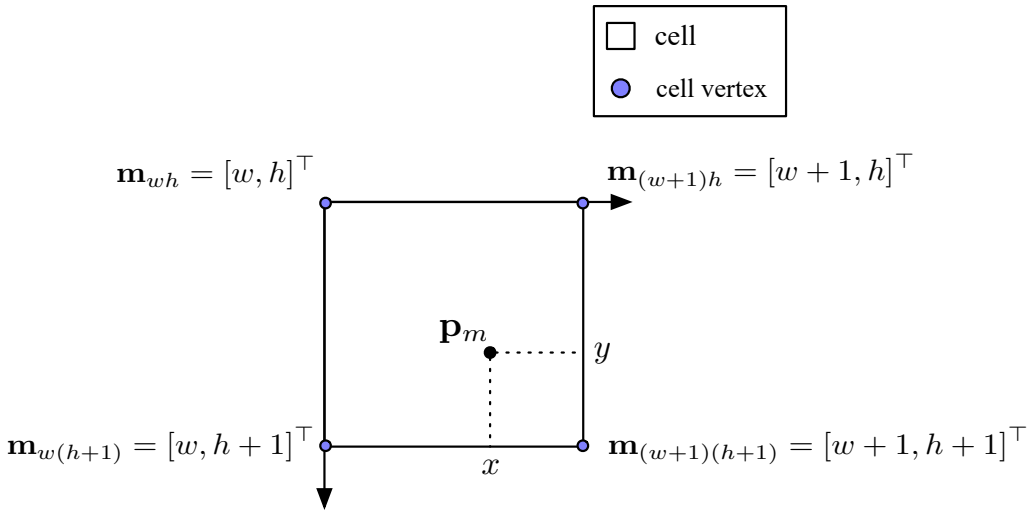


FIGURE 3.1: **Parameterizing the entire map by bilinear interpolation of discrete map $\mathbb{M}$.**

3.1 Problem Formulation

Our approach considers the joint optimization of the robot poses and the occupancy map using information from 2D laser observations (and odometry). In this section, we will explain how the observations from the laser can be linked to the robot poses and the occupancy map to formulate the NLLS problem.

3.1.1 Occupancy Map Representation and State in Optimization

Suppose the environment is discretized into $c_w \times c_h$ grid cells. We use $\mathbf{m}_{wh} = [w, h]^\top$ $(0 \leq w \leq c_w, 0 \leq h \leq c_h)$ to represent the coordinate of a discrete cell vertex in the map. The occupancy value at the cell vertex $\mathbf{m}_{wh}$, denoted as $M(\mathbf{m}_{wh})$, is defined using evidence, which is the natural logarithm of odds (the ratio between the probability of being occupied and the probability of being free)[21, 22, 121]. The occupancy values of all $(c_w+1)\times(c_h+1)$ cell vertices consist of the discrete occupancy map $\mathbb{M} = \{M(\mathbf{m}_{wh})\}_{0\leq w\leq c_w, 0\leq h\leq c_h}$.

To represent the entire environment using a finite number of parameters, we describe the occupancy value at an arbitrary position $\mathbf{p}_m = [x, y]^\top$ on the map using bilinear interpolation of the occupancy values at its four surrounding cell vertices: $\mathbf{m}_{wh}, \mathbf{m}_{(w+1)h}$, $\mathbf{m}_{w(h+1)}, \mathbf{m}_{(w+1)(h+1)}$, as shown in Figure 3.1, i.e.,

$$M(\mathbf{p}_m) = \begin{bmatrix} a_1 b_1, a_0 b_1, a_1 b_0, a_0 b_0 \end{bmatrix} \begin{bmatrix} M(\mathbf{m}_{wh}) \\ M(\mathbf{m}_{(w+1)h}) \\ M(\mathbf{m}_{w(h+1)}) \\ M(\mathbf{m}_{(w+1)(h+1)}) \end{bmatrix} \tag{3.1}$$

in which

$$\begin{aligned} a_0 &= x - w \\ a_1 &= w + 1 - x \\ b_0 &= y - h \\ b_1 &= h + 1 - y. \end{aligned} \tag{3.2}$$

Our method jointly optimizes robot poses and the occupancy map, combining them into the state vector of the proposed optimization problem. Using bilinear interpolation with the discrete occupancy map $\mathbb{M}$, estimating the entire map is equivalent to estimating $\mathbb{M}$. Thus, the map component of the state vector can be expressed as

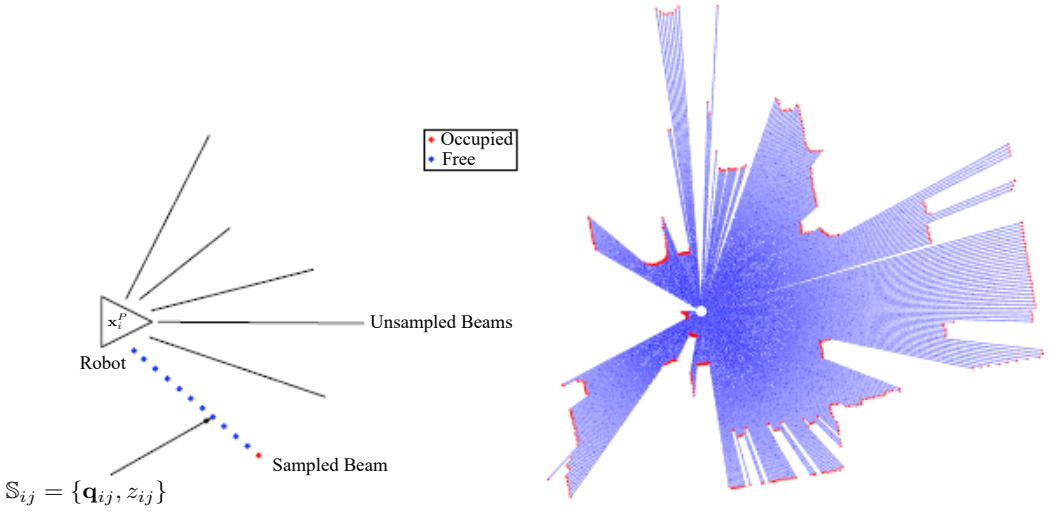
$$\mathbf{x}^M = \left[M(\mathbf{m}_{00}), \cdots, M(\mathbf{m}_{c_w c_h}) \right]^\top. \tag{3.3}$$

We define the $n+1$ robot poses as $\{\mathbf{x}_i^P \triangleq [\mathbf{t}_i^\top, \theta_i]^\top\}_{0 \le i \le n}$, where $\mathbf{t}_i$ is the x-y position of the robot and θ_i is the orientation with the corresponding rotation matrix $\mathbf{R}_i = \begin{bmatrix} \cos(\theta_i), -\sin(\theta_i) \\ \sin(\theta_i), \cos(\theta_i) \end{bmatrix}$. As in most of the SLAM problem formulations, we assume the first robot pose defines the coordinate system, $\mathbf{x}_0^P \triangleq [0,0,0]^\top$, so only the other n robot poses are variables that need to be estimated, thus the pose component of the state vector is represented as

$$\mathbf{x}^P = \left[(\mathbf{x}_1^P)^\top, \cdots, (\mathbf{x}_n^P)^\top \right]^\top. \tag{3.4}$$

Accordingly, the state vector of the proposed optimization problem is

$$\mathbf{x} = \left[(\mathbf{x}^P)^\top, (\mathbf{x}^M)^\top \right]^\top. \tag{3.5}$$



(a) Equidistant Sampling Strategy (b) Observation Points in One Scan

FIGURE 3.2: **Sampling strategy for generating observations from a laser scan.** (a) Equidistant sampling on a beam, with red indicating occupied and blue indicating free states, the distance between two consecutive points is the resolution s. (b) All sampled observation points at a given time step.

In our method, the occupancy map $\mathbb{M}$ is initialized by the Bayesian occupancy mapping method [21] with initially estimated poses (derived from odometry or scan matching) and updated throughout the optimization process.

3.1.2 Scan Points Sampling Strategy

We now introduce our sampling strategy for generating observations from laser scans, which are used in our NLLS formulation.

Each scan data consists of a number of beams. On each beam, the endpoint indicates the presence of an obstacle, while the other points before the endpoint indicate the absence of obstacles. Here, we sample each beam using a fixed resolution s to get the observations, as shown in Figure 3.2(a). Specifically, $\mathbf{q}_{ij} = [x_{q_{ij}}, y_{q_{ij}}]^\top$ denotes the position of jth sampling point at time step i in the local robot/laser coordinate frame and

$$z_{ij} = \ln \frac{p(\mathbf{q}_{ij} \in occ)}{1 - p(\mathbf{q}_{ij} \in occ)} \tag{3.6}$$

denotes the corresponding occupancy value. In the same way as the occupancy map representation described in Section 3.1.1, we also use the evidence to represent the occupancy value here. In our implementation, following [21, 22], we use $p(\mathbf{q}_{ij} \in occ) = 0.7$ for an occupied point (red in Figure 3.2(a)), and use $p(\mathbf{q}_{ij} \in occ) = 0.4$ for a free point (blue in Figure 3.2(a)). Figure 3.2(b) shows an example of all sampled points in one scan. By constant equidistant sampling of all the beams for the scan collected at time step i, a sampling point set

$$\mathbb{S}_i = \{\mathbb{S}_{ij} \triangleq \{\mathbf{q}_{ij}, z_{ij}\}\}_{1 \leq j \leq k_i} \tag{3.7}$$

can be obtained. It should be noted that since the total length of all the beams at different time step i is different, the number of sampling points k_i obtained by the equidistant sampling strategy varies for different time step i.

Suppose there are $n+1$ laser scans collected from robot poses 0 to n, $\mathbb{S} = \{\mathbb{S}_i\}_{0 \leq i \leq n}$ is the available observation information collected at all different robot poses using our sampling strategy and will be used as observations in the proposed NLLS formulation.

3.1.3 Relationship Between Observations and Occupancy Map

In Section 3.1.1, we defined the discrete occupancy map $\mathbb{M}$ as part of the state vector in our optimization problem. Section 3.1.2 detailed the observation generation process. In this section, we explain how the relationship between observations and the occupancy map is established through robot poses, forming the basis of our joint optimization problem.

3.1.3.1 Local to Global Projection

First, the j-th scan point at time step i can be projected to the occupancy map using the robot pose $\mathbf{x}_i^P$, and the projected position on the occupancy map can be calculated by

$$\mathbf{p}_{ij} = \frac{\mathbf{R}_i \mathbf{q}_{ij} + \mathbf{t}_i}{s} \tag{3.8}$$

where s is the resolution of the vertices in the occupancy map $\mathbb{M}$ (the distance between two adjacent cell vertices represents s meters in the real world). Here, we use the same

resolution as that used in generating observations from laser scans in Section 3.1.2. Then, the occupancy value at the projected point $\mathbf{p}_{ij}$ can be obtained using (3.1), expressed as $M(\mathbf{p}_{ij})$.

3.1.3.2 Relationship Between Sampling Points and Occupancy Map w.r.t. Occupancy Values

As outlined in Sections 3.1.1 and 3.1.2, evidence is used to define the occupancy value, where multiple observations of the same cell result in the occupancy values from individual observations being cumulatively added to the cell's total occupancy value [22]. If the robot's poses are accurate and repeated observations of the same cell consistently indicate the same occupancy state, the cell's occupancy value becomes the product of the occupancy value of each observation and the number of times the cell is observed. For a unique coordinate $\mathbf{p}_{ij}$ in (3.8), if both the robot pose $\mathbf{x}_i^P$ and the occupancy map $\mathbb{M}$ are accurate, the occupancy value z_{ij} of its associated sampling point $\mathbb{S}_{ij}$, should closely approximate the occupancy value at $\mathbf{p}_{ij}$, $M(\mathbf{p}_{ij})$, divided by the number of times $\mathbf{p}_{ij}$ is "observed", $N(\mathbf{p}_{ij})$.

Thus, if the number of times the point $\mathbf{p}_{ij}$ is "observed" can be calculated, the relationship between the observations and the state vector (occupancy map and robot poses), can be determined.

3.1.3.3 Hit Map and Hit Number Lookup

We now explain how $N(\mathbf{p}_{ij})$ can be calculated. To quickly query the number of times an arbitrary point is "observed", we need to count the number of times all cell vertices have been observed to form the discrete hit map $\mathbb{N}$ associated with the occupancy map $\mathbb{M}$.

When a sampling scan point is projected into a coordinate by a given robot pose, this coordinate is considered to have been observed once, and then we distribute the hit number "1" of the coordinate to the discretized cell vertices. Since the occupancy value $M(\mathbf{p}_{ij})$ is derived by bilinear interpolation of occupancy values of discrete cell vertices in (3.1), in order to maintain the correspondence between the hit number and the occupancy values,

we distribute this "1" hit to the four surrounding cell vertices by inverse bilinear interpolation. For example, if a sampling point is projected into the center of a cell, then each of the 4 nearby cell vertices gets a hit number of 0.25. In addition, the hit number also accumulates with multiple observations of the same cell vertex, i.e.,

$$[N(\mathbf{m}_{00}), \cdots, N(\mathbf{m}_{c_w c_h})] = \sum_{i=0}^{n} \sum_{j=1}^{k_i} H(\mathbf{p}_{ij}) \tag{3.9}$$

where $H(\cdot)$ is the inverse process of bilinear interpolation. The hit number at all these discrete cell vertices consists of the discrete hit map $\mathbb{N} = \{N(\mathbf{m}_{wh})\}_{0 \leq w \leq c_w, 0 \leq h \leq c_h}$.

After the discrete hit map $\mathbb{N}$ is obtained, the equivalent hit multiplier $N(\mathbf{p}_{ij})$ (representing the number of times $\mathbf{p}_{ij}$ is "observed") for an arbitrary continuous point $\mathbf{p}_{ij}$ can be easily obtained using bilinear interpolation, similar to (3.1).

3.1.4 NLLS Formulation

We now formulate the NLLS problem to jointly optimize the robot poses and the occupancy map. The objective function of the NLLS problem is defined as

$$f(\mathbf{x}) = w_Z f^Z(\mathbf{x}) + w_O f^O(\mathbf{x}) + w_S f^S(\mathbf{x}). \tag{3.10}$$

The objective function consists of the observation term $f^Z(\mathbf{x})$, the smoothing term $f^S(\mathbf{x})$, and the odometry term $f^O(\mathbf{x})$. w_Z, w_S and w_O are their corresponding weights, and we set $w_O = 0$ if there is no odometry information. We now explain the three terms one by one.

3.1.4.1 Observation Term $f^Z(\mathbf{x})$

Based on the relationship between the observations and the occupancy map w.r.t. occupancy values described in 3.1.3, we can formulate the observation term as follows.

Given the observation information $\mathbb{S}$ in (3.7), the observation term in the objective function (3.10) is formulated as

$$f^Z(\mathbf{x}) = \sum_{i=0}^{n} \sum_{j=1}^{k_i} \left\| z_{ij} - F_{ij}^Z(\mathbf{x}) \right\|^2, \tag{3.11}$$

where

$$F_{ij}^Z(\mathbf{x}) = \frac{M(\mathbf{p}_{ij})}{N(\mathbf{p}_{ij})}. \tag{3.12}$$

Here, $\mathbf{p}_{ij}$ represents a coordinate in the map where the j-th sampling scan point at time step i is projected using the robot pose $\mathbf{x}_i^P$, as calculated by (3.8). $N(\mathbf{p}_{ij})$ denotes the equivalent hit multiplier at $\mathbf{p}_{ij}$, as detailed in Section 3.1.3.3.

In (3.11), we suppose the errors of occupancy values of different sampled points in the observations $\mathbb{S}$ are independent and have the same uncertainty. Therefore, the weights on all terms are the same, which is equivalent to setting all the weights as 1. Thus, we use norms instead of weighted norms in equation (3.11).

3.1.4.2 Odometry Term $f^O(\mathbf{x})$

The odometry information $\mathbb{O} = \{\mathbf{o}_i\}_{1 \le i \le n}$ might be available. We assume the odometry input is the relative pose between two consecutive steps. The odometry from robot pose $\mathbf{x}_{i-1}^P$ to pose $\mathbf{x}_i^P$ is expressed as

$$\mathbf{o}_i = \left[(\mathbf{o}_i^t)^\top, o_i^\theta \right]^\top \quad (1 \le i \le n) \tag{3.13}$$

where $\mathbf{o}_i^t$ is the translation part and o_i^θ is the rotation angle part of the odometry. The odometry term can be formulated as

$$\begin{aligned} f^O(\mathbf{x}) &= \sum_{i=1}^{n} \left\| \mathbf{o}_i - F_i^O(\mathbf{x}) \right\|_{\mathbf{\Sigma}_{O_i}^{-1}}^2 \\ &= \sum_{i=1}^{n} \left\| \begin{bmatrix} \mathbf{o}_i^t - \mathbf{R}_{i-1}^\top \left(\mathbf{t}_i - \mathbf{t}_{i-1} \right) \\ \text{wrap}\left(o_i^\theta - \theta_i + \theta_{i-1} \right) \end{bmatrix} \right\|_{\mathbf{\Sigma}_{O_i}^{-1}}^2 \end{aligned} \tag{3.14}$$

in which $\mathbf{\Sigma}_{O_i}$ is the covariance matrix representing the uncertainty of $\mathbf{o}_i$, and $\text{wrap}(\cdot)$ wraparounds the rotation angle to $(-\pi, \pi]$.

3.1.4.3 Smoothing Term $f^S(\mathbf{x})$

It can be easily found out that minimizing the objective function with only the observation term (and the odometry term) is not easy since there are a large number of local minima. Especially when the initial robot poses are far away from the global minimum, it is very difficult for an optimizer to converge to the correct solution.

In order to enlarge the region of attraction and develop an algorithm that is robust to initial values, we introduce a smoothing term. The smoothing term requires the occupancy values of nearby cell vertices to be close to each other, thus resulting in the occupancy map being smoother for derivative calculation. In our case, based on the derivative calculation method we use (see Appendix A.1), we penalize the difference between the occupancy value of each cell vertex and the occupancy values of the two neighboring cell vertices to its right and below, i.e.,

$$\begin{aligned} f^S(\mathbf{x}) &= \left\| F^S(\mathbf{x}) \right\|^2 \\ &= \sum_{w=0}^{c_w-1} \sum_{h=0}^{c_h-1} \left\| \begin{bmatrix} M(\mathbf{m}_{wh}) - M(\mathbf{m}_{(w+1)h}) \\ M(\mathbf{m}_{wh}) - M(\mathbf{m}_{w(h+1)}) \end{bmatrix} \right\|^2 \\ &+ \sum_{h=0}^{c_h-1} \left\| M(\mathbf{m}_{c_w h}) - M(\mathbf{m}_{c_w (h+1)}) \right\|^2 \\ &+ \sum_{w=0}^{c_w-1} \left\| M(\mathbf{m}_{w c_h}) - M(\mathbf{m}_{(w+1) c_h}) \right\|^2, \end{aligned} \tag{3.15}$$

where the second and third terms are used to handle cell vertices located in the bottom row and the rightmost column. It should be noted that $F^S(\mathbf{x})$ is a linear function of $\mathbf{x}^M$ in the state. The coefficient matrix is constant and can be calculated prior to the optimization. For more details, please refer to Appendix A.4.

3.2 Iterative Solution to the NLLS Formulation

In Section 3.1, we introduced our NLLS formulation for the joint poses and occupancy map optimization problem. In this section, we provide the details of a GN based algorithm for solving the NLLS problem.

In the equation below, we assume the odometry inputs are available. Let

$$\begin{aligned} F(\mathbf{x}) &= \left[\cdots, z_{ij} - F_{ij}^{Z}(\mathbf{x}), \cdots, (\mathbf{o}_i - F_i^{O}(\mathbf{x}))^\top, \cdots, F^{S}(\mathbf{x})^\top\right]^\top \\ \mathbf{W} &= \ \mathrm{diag}(\cdots, w_Z, \cdots, w_O \boldsymbol{\Sigma}_{O_i}^{-1}, \cdots, w_S, \cdots) \end{aligned} \tag{3.16}$$

combine all the error functions and the weights of the three terms in (3.10). Then, the NLLS problem in (3.10) seeks $\mathbf{x}$ such that

$$f(\mathbf{x}) = \|F(\mathbf{x})\|_{\mathbf{W}}^2 = F(\mathbf{x})^\top \mathbf{W} F(\mathbf{x}) \tag{3.17}$$

is minimized.

A solution to (3.17) can be obtained iteratively by starting with an initial guess $\mathbf{x}(0)$ and updating with $\mathbf{x}(k+1) = \mathbf{x}(k) + \boldsymbol{\Delta}(k)$. The update vector $\boldsymbol{\Delta}(k) = \left[\boldsymbol{\Delta}^P(k)^\top, \boldsymbol{\Delta}^M(k)^\top\right]^\top$ is the solution to

$$\mathbf{J}^\top \mathbf{W} \mathbf{J} \boldsymbol{\Delta}(k) = -\mathbf{J}^\top \mathbf{W} F(\mathbf{x}(k)) \tag{3.18}$$

where $\mathbf{J}$ is the linear mapping represented by the Jacobian matrix $\partial F / \partial \mathbf{x}$ evaluated at $\mathbf{x}(k)$.

The iterative method for solving the proposed NLLS problem is shown in Algorithm 1, in which τ_k and τ_Δ represent the thresholds of iteration number k and the incremental vector $\boldsymbol{\Delta}$. Unlike the standard GN iterative method, the hit map needs to be additionally recalculated after updating the poses in each iteration. With this approach, the implicit data association is established at each iteration and updated during the optimization.

Since the robot poses and the occupancy map are optimized simultaneously, the Jacobian $\mathbf{J}$ in (3.18) is very important and quite different from those used in the traditional SLAM algorithms. More details of the Jacobians are described in appendices.

3.3 Multi-resolution Joint Optimization Strategy

Algorithm 1 provides a solution to our NLLS problem (3.10) to jointly optimize the poses and the occupancy map. However, directly using Algorithm 1 with the high-resolution

Algorithm 1: Our Joint Poses and Occupancy Map Optimization Algorithm

Params: Threshold τ_k, τ_Δ, weight matrix $\mathbf{W}$, resolution s
Input: Observations $\mathbb{S}$, odometry $\mathbb{O}$, and initial poses $\mathbf{x}^P(0)$
Output: Optimized poses $\hat{\mathbf{x}}^P$ and optimized map $\hat{\mathbf{x}}^M$

Function `FirstStage`($\mathbf{x}^P(0)$, $\mathbb{S}$, $\mathbb{O}$, τ_k, τ_Δ, s, $\mathbf{W}$)**:**
  Initialize $\mathbf{x}^M(0)$ and $\mathbb{N}(0)$ using $\mathbf{x}^P(0)$ and $\mathbb{S}$
  Pre-calculate smoothing term coefficient $\mathbf{A}$ using (A.8)
  for $k = 0$; $k <= \tau_k$ & $\|\boldsymbol{\Delta}(k)\|^2 >= \tau_\Delta$; $k++$ **do**
    Function `OccupancyGN`($\mathbf{x}^M(k)$, $\mathbb{N}(k)$, $\mathbf{x}^P(k)$, $\mathbb{S}$, $\mathbb{O}$, $\mathbf{A}$, $\mathbf{W}$)**:**
      Calculate gradient $\nabla\mathbf{x}^M(k)$ of $\mathbf{x}^M(k)$
      Calculate $\mathbf{J}$, as described in appendices
      Evaluate $F(\mathbf{x})$ at $\mathbf{x}^P(k)$ and $\mathbf{x}^M(k)$
      Solve $\mathbf{J}^\top\mathbf{W}\mathbf{J}\boldsymbol{\Delta}(k) = -\mathbf{J}^\top\mathbf{W}F(\mathbf{x})$, where $\boldsymbol{\Delta}(k) = [\boldsymbol{\Delta}^P(k)^\top, \boldsymbol{\Delta}^M(k)^\top]^\top$
      Update $\mathbf{x}^P(k+1) = \mathbf{x}^P(k) + \boldsymbol{\Delta}^P(k)$ and $\mathbf{x}^M(k+1) = \mathbf{x}^M(k) + \boldsymbol{\Delta}^M(k)$
      Recalculate $\mathbb{N}(k+1)$ using $\mathbf{x}^P(k+1)$ and $\mathbb{S}$
      `return`($\mathbf{x}^P(k+1)$, $\mathbf{x}^M(k+1)$)
    End Function
  end
  $\hat{\mathbf{x}}^P \Leftarrow \mathbf{x}^P(k)$, $\hat{\mathbf{x}}^M \Leftarrow \mathbf{x}^M(k)$
  `return`($\hat{\mathbf{x}}^P$, $\hat{\mathbf{x}}^M$)
End Function

map is time-consuming and requires an accurate initial value of robot poses [15], which is challenging to obtain. To overcome these limitations, we propose a multi-resolution joint optimization strategy in this section.

3.3.1 Discussion on Map Resolution in Optimization

The resolution of the occupancy map has a significant impact on the optimization results since $\mathbf{x}^M$ is part of the state vector in our NLLS formulation (3.10).

A high-resolution map enables precise relationships between observations and occupancy values of projected points. However, it results in a dramatic increase in the optimization problem's size, raising computational costs. Additionally, in a high-resolution map, occupancy values in adjacent cell vertices may exhibit sharper variations compared to those in a low-resolution map, leading to noisy gradients when poor initial robot poses are used. Even with the introduction of the smoothing term, the use of a high-resolution map may cause poor convergence of Algorithm 1.

A low-resolution map provides advantages in faster computation and reduced memory usage. Moreover, gradients are less sensitive to pose accuracy. With our occupancy map representation and smoothing term, these advantages enable the algorithm to quickly converge to a reasonable solution, even with poor initial robot poses. However, low resolution may cause inaccurate links between observations and occupancy values near boundaries, preventing the optimization from achieving greater accuracy.

To combine the advantages of different resolution map representations, we propose a multi-resolution strategy to optimize the occupancy values of different resolution cell vertices together with robot poses at various stages. Unlike the conventional coarse-to-fine scheme, in the second stage of our strategy, we use the selected high-resolution map that only includes high-resolution cell vertices possibly in need of further optimization instead of the full high-resolution map. Optimizing only those selected high-resolution cell vertices further improves the efficiency of our algorithm.

3.3.2 Multi-resolution Joint Optimization Strategy

Firstly, we obtain low-resolution observations $\mathbb{S}^l = \{\mathbb{S}_i^l\}_{0 \le i \le n}$ by down-sampling from the high-resolution observations $\mathbb{S}^h = \{\mathbb{S}_i^h\}_{0 \le i \le n}$, which are obtained by the equal sampling strategy described in Section 3.1.2 with a sampling distance s^h. Here, we set the map resolution and sampling resolution to be the same. Therefore, the low resolution $s^l = r \times s^h$, where r is the resolution ratio between the low-resolution map and the high-resolution map. The size of the low-resolution map $\mathbb{M}^l$ is $(c_w + 1) \times (c_h + 1)$.

Initialized by the odometry inputs or scan matching, we perform Algorithm 1 to quickly obtain relatively accurate poses. The state vector in the first stage is $\mathbf{x}^l = \left[\mathbf{x}^{P\top}, \mathbf{x}^{lM\top}\right]^\top$, where $\mathbf{x}^{lM}$ includes all occupancy values at the cell vertices of the low-resolution map $\mathbb{M}^l$. In this stage, the hit map, observation information, coefficient matrix, weight matrix, and resolution are represented as $\mathbb{N}^l, \mathbb{S}^l, \mathbf{A}^l$, $\mathbf{W}^l$, and s^l, respectively.

In the first stage of optimization, the low-resolution occupancy map reduces both the dimension of $\mathbf{x}^{lM}$ and the number of observations in $\mathbb{S}^l$. Since the occupancy values at cell vertices change relatively gradually in the low-resolution map, the directions of the map's

Algorithm 2: Finding the Selected High-resolution Map and Corresponding Observations

Params: Resolution s^h, selection distance d, convolution kernel size q
Input: Observations $\mathbb{S}^h$, and poses $\hat{\mathbf{x}}^{\tilde{P}}$ from the first stage using Algorithm 1
Output: Observations $\mathbb{S}^s$ and map part of the state vector in the second stage $\mathbf{x}^{sM}$

Function `Selection(`$\hat{\mathbf{x}}^{\tilde{P}}$`,` $\mathbb{S}^h$`,` s^h`,` d`,` q`):`
  Build a full high-resolution map $\mathbb{M}^h$ using $\hat{\mathbf{x}}^{\tilde{P}}$ and $\mathbb{S}^h$
  Calculate the binary map $\mathbb{B}$ using $\mathbb{M}^h$
  Calculate the convolved map $\mathbb{C}$ with kernel size q
  Calculate the set $\mathbb{I}^h$, which includes the indices of all boundary vertices in $\mathbb{M}^h$, using $\mathbb{C}$
  Calculate the set $\mathbb{I}^s$, which includes the indices of all selected vertices, using $\mathbb{I}^h$ and d
  Define the map part of the state vector in the second stage $\mathbf{x}^{sM}$ and the selected high-resolution map $\mathbb{M}^s$ by $\mathbb{I}^s$
  Find observations $\mathbb{S}^s$ for $\mathbb{M}^s$ using the set $\mathbb{I}^s$, $\mathbb{S}^h$ and $\hat{\mathbf{x}}^{\tilde{P}}$
  `return(`$\mathbb{S}^s$`,` $\mathbf{x}^{sM}$`)`
End Function

gradients are closer to the correct ones when the poses are initialized by odometry inputs or scan matching, making it easier for Algorithm 1 to converge to a relatively good result quickly.

After the first stage, we use Algorithm 2 to select the cell vertices that need to be further optimized to compose the selected high-resolution map $\mathbb{M}^s$ and find their corresponding observations $\mathbb{S}^s$. Details are described in Section 3.3.3.

In the second stage, the state vector is represented as $\mathbf{x}^s = \left[\mathbf{x}^{P\top}, \mathbf{x}^{sM\top}\right]^\top$ where $\mathbf{x}^{sM}$ includes all occupancy values at cell vertices of the selected high-resolution map $\mathbb{M}^s$. We perform Algorithm 3 using poses obtained from the first stage as initial guesses and observations $\mathbb{S}^s$ to refine poses. The NLLS optimization problem in the second stage can be formulated similarly as (3.17). Additionally, the differences in the Jacobian calculation between Algorithm 1 and Algorithm 3 are described in Appendix A.5.

The full multi-resolution joint optimization strategy is outlined in Algorithm 4.

3.3.3 Selected High-resolution Map and Observations

After the first stage optimization using the low-resolution map $\mathbb{M}^l$, the robot poses $\hat{\mathbf{x}}^{\tilde{P}}$ become relatively accurate. Subsequently, the full high-resolution map $\mathbb{M}^h$, with dimensions $(r * c_w + 1) \times (r * c_h + 1)$, is built using the Bayesian occupancy mapping method [21], based

Algorithm 3: The Algorithm for the Second Stage of the Multi-resolution Joint Optimization Strategy

Params: Threshold τ_k^s, τ_Δ^s, weight matrix $\mathbf{W}^s$, resolution s^h
Input: Observations $\mathbb{S}^s$, odometry $\mathbb{O}$, and poses $\hat{\mathbf{x}}^{\tilde{P}}$ from the first stage using Algorithm 1
Output: Optimal poses $\hat{\mathbf{x}}^P$ and map $\hat{\mathbf{x}}^{sM}$

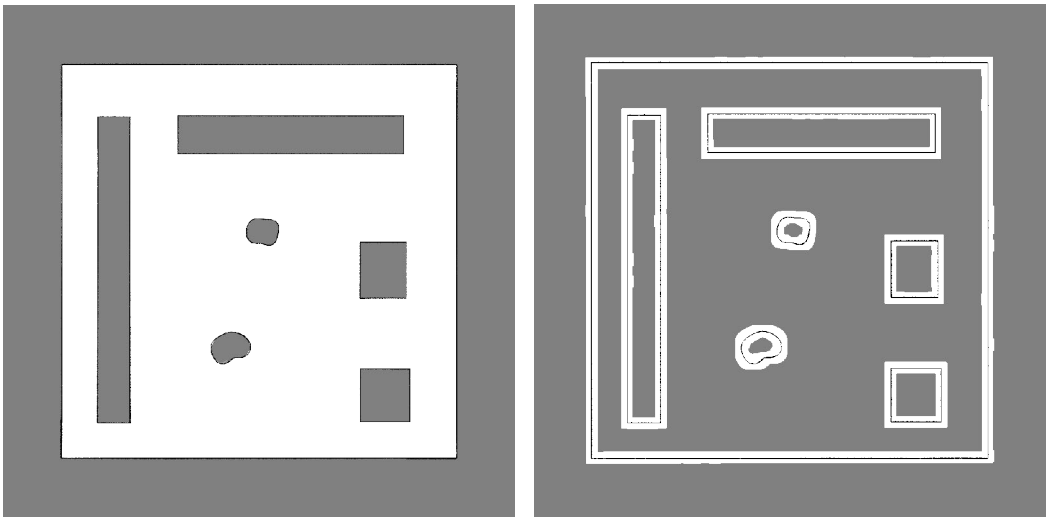
$\mathbf{x}^P(0) \Leftarrow \hat{\mathbf{x}}^{\tilde{P}}$
Function `SecondStage(`$\mathbf{x}^P(0)$, $\mathbb{S}^s$, $\mathbb{O}$, τ_k^s, τ_Δ^s, s^h, $\mathbf{W}^s$`):`
  Initialize $\mathbf{x}^{sM}(0)$ and $\mathbb{N}^s(0)$ using $\mathbb{S}^s$ and $\mathbf{x}^P(0)$
  Pre-calculate smoothing term coefficient matrix $\mathbf{A}^s$
  for $k = 0$; $k <= \tau_k^s$ & $\|\mathbf{\Delta}(k)\|^2 >= \tau_\Delta^s$; $k++$ **do**
    $\mathbf{x}^P(k+1), \mathbf{x}^{sM}(k+1) \leftarrow$ `OccupancyGN(`$\mathbf{x}^{sM}(k)$, $\mathbb{N}^s(k)$, $\mathbf{x}^P(k)$, $\mathbb{S}^s$, $\mathbb{O}$, $\mathbf{A}^s$, $\mathbf{W}^s$`)`
  end
  $\hat{\mathbf{x}}^P \Leftarrow \mathbf{x}^P(k)$, $\hat{\mathbf{x}}^{sM} \Leftarrow \mathbf{x}^{sM}(k)$
  `return(`$\hat{\mathbf{x}}^P$, $\hat{\mathbf{x}}^{sM}$`)`
End Function

Algorithm 4: Multi-resolution Joint Optimization Strategy

Params: Threshold τ_k^l, τ_Δ^l, τ_k^s, τ_Δ^s, weight matrix $\mathbf{W}^l$, $\mathbf{W}^s$, ratio of resolutions r, resolution s, selection distance d, convolution kernel size q
Input: Observations $\mathbb{S}^h$, odometry $\mathbb{O}$
Output: Optimal poses $\hat{\mathbf{x}}^P$ and map $\hat{\mathbf{x}}^{sM}$

$\mathbb{S}^l \leftarrow$ `DownSampling(`$\mathbb{S}^h$, r`)`
if $w_O \neq 0$ **then**
  $\mathbf{x}^P(0) \leftarrow$ `InitializePose(`$\mathbb{O}$`)`
else
  $\mathbf{x}^P(0) \leftarrow$ `ScanMatching(`$\mathbb{S}^l$`)`
end
$\hat{\mathbf{x}}^{\tilde{P}}, \hat{\mathbf{x}}^{lM} \leftarrow$ `FirstStage(`$\mathbf{x}^P(0)$, $\mathbb{S}^l$, $\mathbb{O}$, τ_k^l, τ_Δ^l, s^l, $\mathbf{W}^l$`)`
$\mathbb{S}^s, \mathbf{x}^{sM} \leftarrow$ `Selection(`$\hat{\mathbf{x}}^{\tilde{P}}$, $\mathbb{S}^h$, s^h, d, q`)`
$\hat{\mathbf{x}}^P, \hat{\mathbf{x}}^{sM} \leftarrow$ `SecondStage(`$\hat{\mathbf{x}}^{\tilde{P}}$, $\mathbb{S}^s$,$\mathbb{O}$, τ_k^s, τ_Δ^s, s^h, $\mathbf{W}^s$`)`

on observations $\mathbb{S}^h$ and poses $\hat{\mathbf{x}}^{\tilde{P}}$. In this case, most cell vertices of $\mathbb{M}^h$ are considered stable in terms of occupancy state. Semantically, these stable cell vertices have the same occupancy state as the surrounding cell vertices (typically free or unknown cells). This characteristic leads to map gradients near zero at these stable cell vertices. In contrast, the cell vertices that require further updates are typically located at the edges of objects, where the occupancy values significantly differ from those of surrounding cell vertices. Therefore, the gradient at these cell vertices is larger. An example illustrating this is shown in Figure 3.3(b), where the selected area (in white and black) is clearly distinct from the stable area (in gray). Based on this idea, we propose a strategy to select the cell vertices located around the boundaries to compose the selected high-resolution map $\mathbb{M}^s$,



(a) Full High-resolution Map

(b) Selected High-resolution Map (In White and Black)

FIGURE 3.3: **An example of the selected high-resolution map from a full high-resolution map in a simulation dataset.** (a) The full high-resolution map generated using poses from the first-stage optimization and scans, forming the basis for selection. (b) The recolored selected high-resolution map: gray marks dropped (stable) areas, white and black denote selected areas, with black highlighting obstacle boundaries.

which is used in the second stage of optimization.

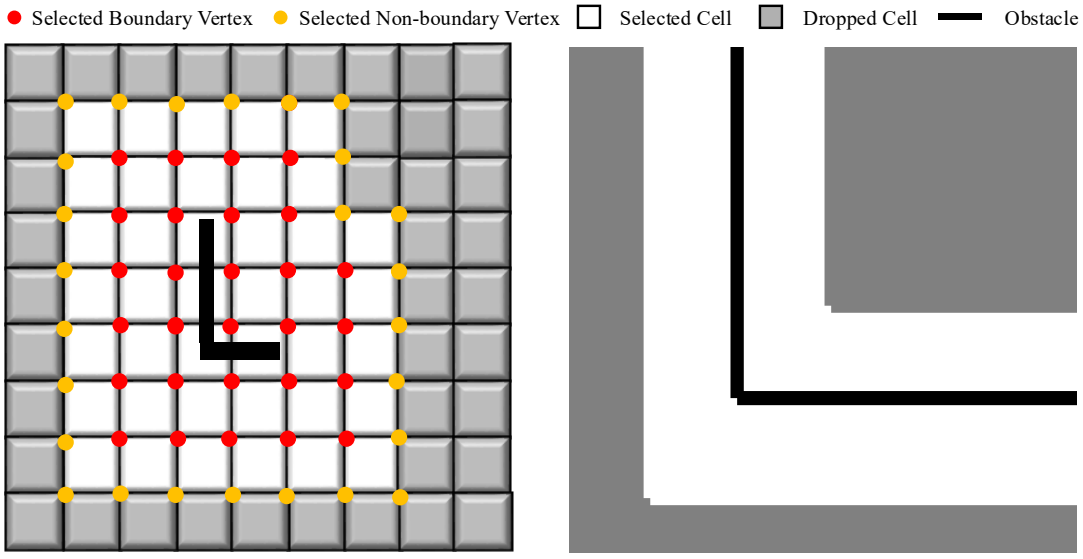
Firstly, we identify cell vertices located at the edges of objects by performing mean-value convolution of the full high-resolution map $\mathbb{M}^h$. Specifically, we calculate a binary map $\mathbb{B} = \{B(\mathbf{m}_{id})\}$ by binarizing $\mathbb{M}^h$ as

$$B(\mathbf{m}_{id}) = \begin{cases} 1, & M^h(\mathbf{m}_{id}) \geq \tau_{occupied} \\ 0, & M^h(\mathbf{m}_{id}) < \tau_{occupied} \end{cases}, \tag{3.19}$$

where $\mathbf{m}_{id}$ represents a cell vertex, and $\tau_{occupied}$ is the threshold used to classify a cell vertex as occupied or free. A mean-value convolution kernel $\mathbf{K}$ is defined as

$$\mathbf{K} = \frac{1}{q^2} \cdot \mathbf{1}_{q \times q} \tag{3.20}$$

where $\mathbf{1}_{q \times q}$ represents a $q \times q$ matrix of ones. The convolved map $\mathbb{C} = \{C(\mathbf{m}_{id})\}$ is then derived by convolving $\mathbb{B}$ with $\mathbf{K}$, where $C(\mathbf{m}_{id})$ indicates whether the $q \times q$ cell vertices around $\mathbf{m}_{id}$ are all in the same occupancy state. Compared to other edge detection



(a) Cell Vertices Selection Strategy (b) Part of Selected High-resolution Map

FIGURE 3.4: **An illustration of the cell vertices selection strategy and a selected high-resolution map from a simulation dataset.** In (a), selected cell vertices are marked in red and yellow, with their indices forming the index set $\mathbb{I}^s$.

methods like Sobel [122] and Canny [123], this conservative method more reliably selects cell vertices that may require further optimization.

Using this method, the set of indices for all boundary cell vertices in the high-resolution map is defined as

$$\mathbb{I}^h = \{id | 0 < C(\mathbf{m}_{id}) < 1\}. \tag{3.21}$$

The cell vertices indexed in $\mathbb{I}^h$ are marked in red in Figure 3.4(a).

To account for pose uncertainties from the first stage, the selection is expanded to include cell vertices within a distance d from all boundary cell vertices. The indices of the selected cell vertices in the high-resolution map form the set $\mathbb{I}^s$, illustrated in Figure 3.4(a), where the selected cell vertices are highlighted in red and yellow with $d = 1$. An example of a selected high-resolution map from a simulation dataset is shown in Figure 3.4(b).

Consequently, the map component of the state vector in the second stage is expressed as

$$\mathbf{x}^{sM} = \left[\cdots, M^h(\mathbf{m}_{wh}), \cdots\right]^\top, \quad wh \in \mathbb{I}^s. \tag{3.22}$$

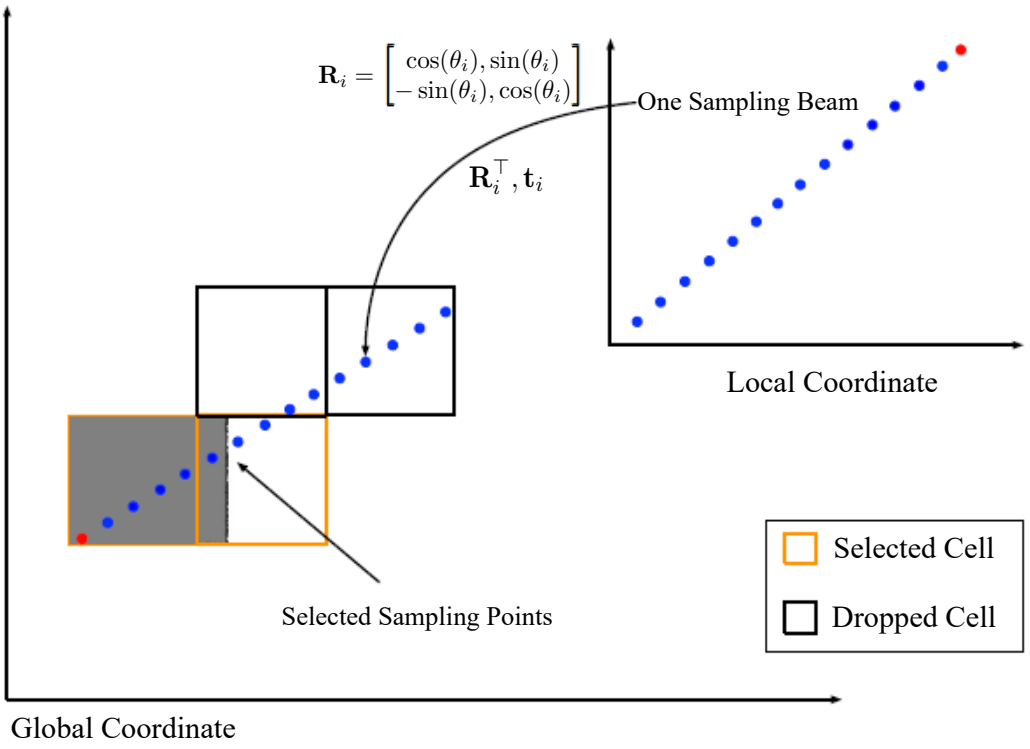


FIGURE 3.5: **An example of the selected sampling points of a beam at time step** i, where points projected onto the selected cells are chosen.

Next, we select observations to optimize $\mathbf{x}^{sM}$. Cells surrounded by vertices with indices in $\mathbb{I}^s$ are designated as selected cells, shown in white in Figure 3.4(a) and Figure 3.4(b). Subsequently, sampling point selection is carried out, as illustrated in Figure 3.5. Specifically, sampling points in $\mathbb{S}^h$ are first projected onto the global coordinate system using the poses optimized in the first stage. All sampling points located on the selected cells are then included to form the set $\mathbb{S}^s$.

3.4 Experimental Results

In this section, we evaluate the proposed algorithm on several datasets and compare its performance with Cartographer [8], the current state-of-the-art algorithm, which significantly outperforms other methods such as Hector-SLAM [14] and Karto-SLAM [67]. To ensure fair comparisons, we adjust some parameters in Cartographer based on the sensor configurations of the respective datasets for optimal performance.

The dataset parameters are summarized in Table 3.1. For practical datasets, Deutsches Museum b0 is one of the Cartographer demo datasets collected at the Deutsches Museum. The Car Park [94] dataset is gathered in an underground car park, while C5 is collected in a factory environment using a Hokuyo UTM-30LX laser scanner. Consistent map resolutions s are applied across all methods to display the map results, with ratio r set to 10 for all simulation experiments and 5 for all practical experiments unless stated otherwise. For each dataset, 20% of scans and corresponding poses are uniformly selected as key frames for the key frame option in our method.

TABLE 3.1: Parameters of Datasets.

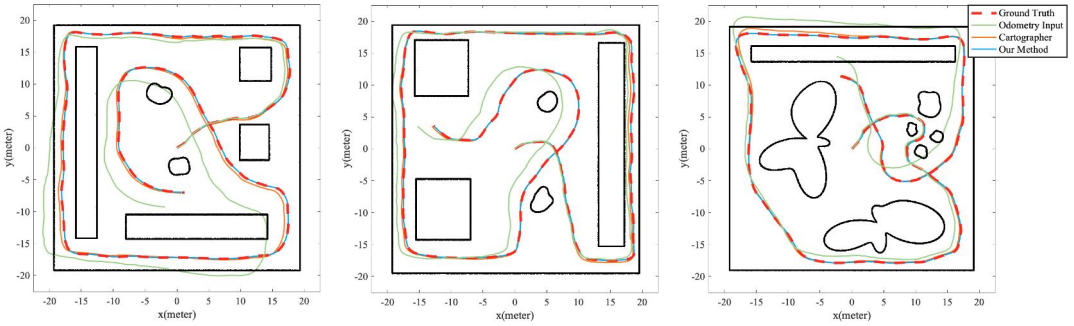
Dataset	No. Scans	Duration	Map Size	Odometry	Resolution
Simulation 1	3640	117 s	50 m × 50 m	yes	0.05 m
Simulation 2	3720	121 s	50 m × 50 m	yes	0.05 m
Simulation 3	2680	83 s	50 m × 50 m	yes	0.05 m
Car Park	1642	164 s	50 m × 40 m	yes	0.1 m
C5	3870	136 s	50 m × 40 m	yes	0.1 m
Museum b0	5522	152 s	85 m × 95 m	no	0.1 m

To ensure fair comparisons, we use an identical number of poses (synchronizing the poses from the results with the ground truth poses using timestamps) and their corresponding observations to generate results for visualization and quantitative evaluation across all compared methods, with the exception of our method which employs keyframes. Furthermore, the same occupancy mapping algorithm is applied consistently across all approaches to produce the OGM results for comparison.

3.4.1 Simulation Experiments

We use three different simulation environments with varying levels of nonlinearity and nonconvex obstacles to design three different simulation experiments. Since Cartographer needs a high-frequency scanning rate to ensure the good performance of scan matching, while our approach performs well for scan data with low scanning frequency, only 10% of scans listed in Table 3.1 are used in our method.

We utilize the open-source 2D LiDAR simulator from [94] to generate simulated datasets. Each scan includes 1081 laser beams spanning angles from -135 degrees to 135 degrees,



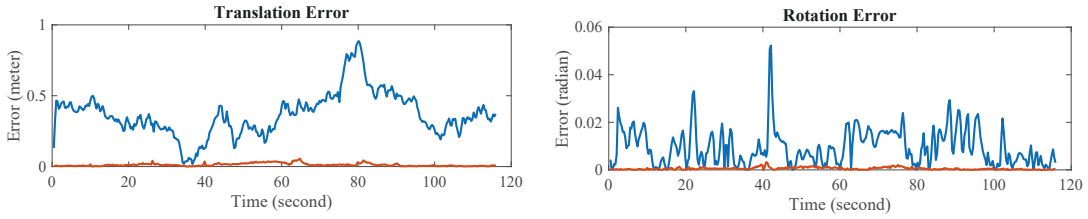
(a) Simulation 1 (b) Simulation 2 (c) Simulation 3

FIGURE 3.6: **Simulation environments and robot trajectory results.** (a), (b), and (c) show the simulation environments (the black lines indicate the obstacles in the scene) and the trajectories of ground truth, odometry inputs, Cartographer [8], and our approach for one dataset in each of the three simulation environments.

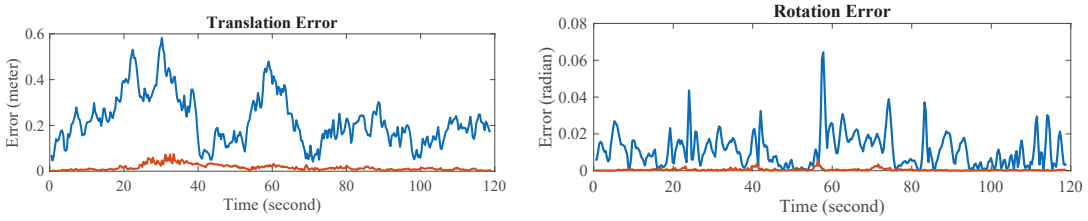
mimicking the specifications of a Hokuyo UTM-30LX laser scanner. To emulate real-world data acquisition, random Gaussian noise with zero mean and standard deviation of 0.02 m is added to each beam of the simulated scan data. Similarly, zero-mean Gaussian noise is introduced to the odometry inputs derived from the ground truth poses, with standard deviation of 0.04 m for x-y and 0.003 rad for orientation. Five datasets with different noise realizations are generated for each simulation environment.

The trajectory results of our method and Cartographer, compared to ground truth and odometry, are shown in Figure 3.6. It is evident that our trajectories align more closely with the ground truth, particularly in areas with significant rotational movements. Figure 3.7 illustrates translation and rotation errors over time, demonstrating that our method consistently achieves substantially smaller errors compared to Cartographer.

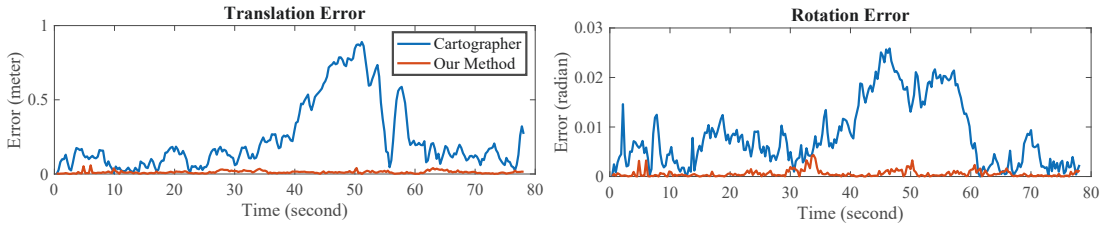
We performed quantitative and qualitative comparisons of pose estimation errors using all fifteen datasets from Simulations 1, 2, and 3. Table 3.2 presents, in order, the quantitative results for odometry inputs, Cartographer, the first stage of our method (Algorithm 1 with low-resolution) using all frames, our method using all frames, and our method using key frames. Metrics such as mean absolute error (MAE) and root mean squared error (RMSE) evaluate translation errors (in meters) and rotation errors (in radians). Our method consistently achieves the best performance across all metrics, significantly outperforming Cartographer even when using only key frames or the first stage. Figure 3.8(a) to Figure 3.8(e) further illustrates OGMs and point cloud maps generated using poses from



(a) Simulation 1



(b) Simulation 2



(c) Simulation 3

FIGURE 3.7: **Comparison of translation and rotation errors at different time**

the ground truth, Cartographer, and the three options of our method. The maps produced by our method exhibit noticeably clearer boundaries, demonstrating its ability to jointly optimize robot poses and occupancy maps for more accurate results.

From the results of our first stage shown in Figure 3.8(c), it is evident that further optimization is needed at the edges of objects. This is due to sampling points with different occupancy values being projected onto the coarse grid cells at object boundaries, causing inaccurate data associations. These results highlight the necessity of the second stage in our multi-resolution strategy (Algorithm 3) to improve accuracy. Additionally, as shown in Figure 3.8(c), the non-edge areas (stable areas) of the OGMs are well optimized, supporting the fact that including occupancy cell vertices of non-edge areas in the state variables for further optimization is unnecessary.

For a quantitative comparison of the occupancy maps, we apply the same threshold across

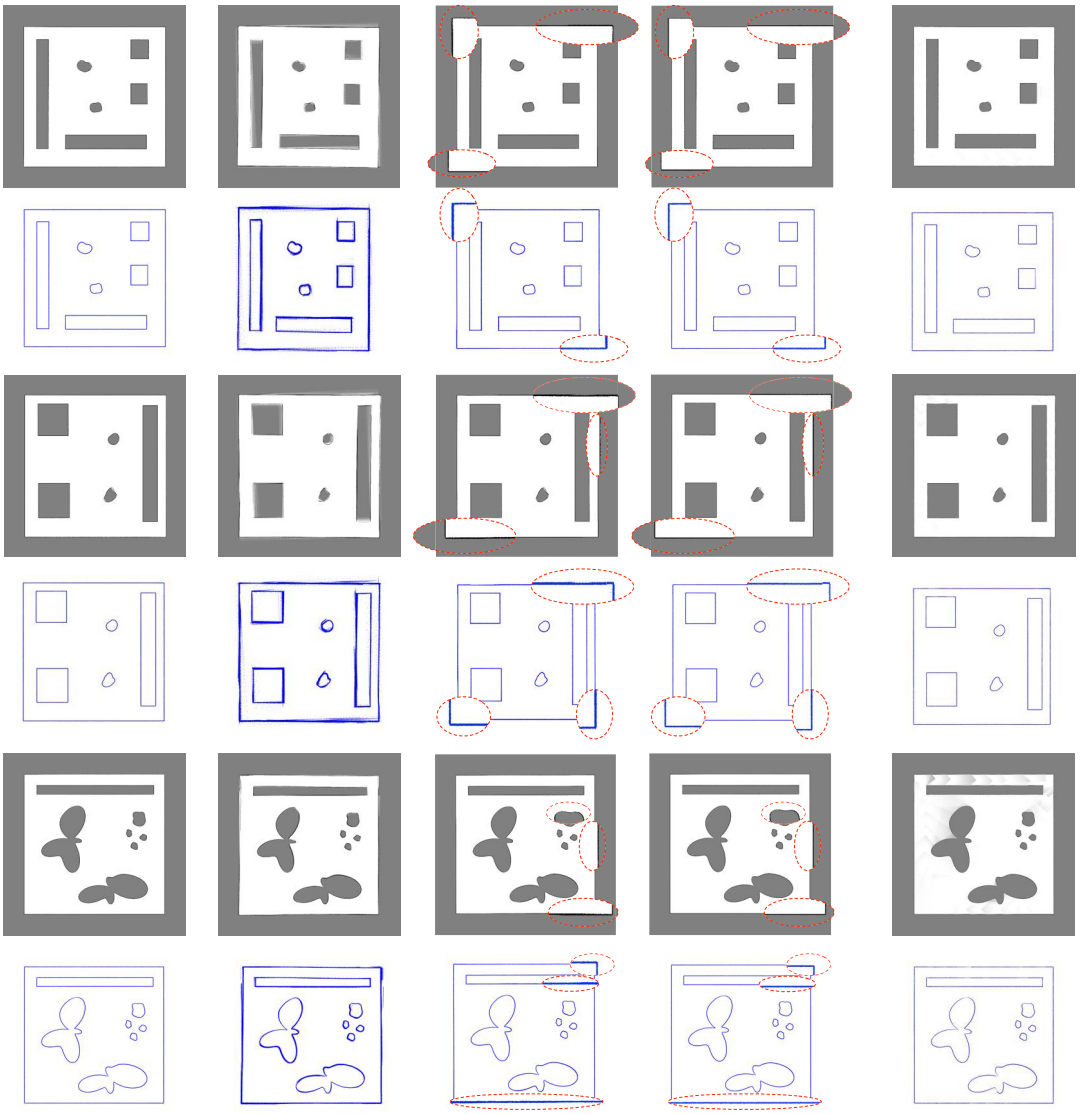


FIGURE 3.8: **The OGMs and point cloud maps generated from ground truth poses and different approaches for each simulation dataset.** The areas marked with red dots highlight where our method outperforms the results of the first-stage optimization alone.

all methods to convert occupancy values into occupancy states. The mapping problem is treated as a classification task, categorizing each grid cell as free, occupied, or unknown. We assess performance using Area under the ROC curve (AUC) [124] and precision, with ground truth labels generated from the occupancy map based on ground truth poses. To ensure a fair comparison, all unknown cells are excluded from this evaluation, as AUC is a binary classification metric [124]. Table 3.6 presents the results, showing that our

TABLE 3.2: Quantitative Comparison of Robot Pose Errors in Simulations.

	Odom	Carto	Ours (First)	Ours (All)	Ours (Key)
Simulation 1					
MAE / Trans (m)	0.78270	0.25336	0.02206	**0.00640**	**0.01024**
MAE / Rot (rad)	0.04912	0.01394	0.00098	**0.00060**	**0.00084**
RMSE / Trans (m)	0.98404	0.29920	0.02680	**0.00974**	**0.01430**
RMSE / Rot(rad)	0.05506	0.01562	0.00162	**0.00102**	**0.00126**
Simulation 2					
MAE / Trans (m)	0.80544	0.11914	0.03224	**0.00858**	**0.01082**
MAE / Rot (rad)	0.02538	0.00666	0.00220	**0.00062**	**0.00096**
RMSE / Trans (m)	0.97152	0.14810	0.04188	**0.01198**	**0.01244**
RMSE / Rot (rad)	0.02936	0.00916	0.00220	**0.00104**	**0.00178**
Simulation 3					
MAE / Trans(m)	0.75352	0.14262	0.02624	**0.00726**	**0.00998**
MAE / Rot (rad)	0.05180	0.00682	0.00164	**0.00058**	**0.00090**
RMSE / Trans (m)	0.96866	0.18782	0.03238	**0.00952**	**0.01338**
RMSE / Rot (rad)	0.05926	0.00914	0.00204	**0.00088**	**0.00134**

Red and **blue** indicate the best and second best results, respectively.

TABLE 3.3: OGM Precision of Our Method Using All Frames, Our Method Using Key Frames, and Cartographer – Simulation 1.

		Ground Truth		
		Unknown	Free	Occupied
Our Method (All Frames)	Unknown	**99.798%**	0.022%	5.436%
	Free	0.020%	**99.938%**	2.334%
	Occupied	0.182%	0.040%	**92.230%**
Our Method (Key Frames)	Unknown	**99.598%**	0.094%	13.562%
	Free	0.072%	**99.824%**	3.142%
	Occupied	0.330%	0.082%	**83.296%**
Cartographer	Unknown	95.616%	1.290%	30.053%
	Free	2.822%	97.678%	53.280%
	Occupied	1.562%	1.032%	16.667%

Predicted

method using all frames achieves the highest performance in both metrics. Even with only key frames, our method surpasses Cartographer. A key factor resulting in Cartographer's lower mapping quality is its lack of a batch optimization method to address errors during submap construction. Although its scan-to-map matching approach reduces cumulative errors more effectively than scan-to-scan matching, its accuracy still falls short compared to our algorithm. Cartographer performs pose graph optimization to adjust the coordinate frames of submaps only when loop closure is detected, leaving errors within the submaps

TABLE 3.4: OGM Precision of Our Method Using All Frames, Our Method Using Key Frames, and Cartographer – Simulation 2.

		Ground Truth		
		Unknown	Free	Occupied
Predicted				
Our Method (All Frames)	Unknown	**99.846%**	0.016%	6.434%
	Free	0.010%	**99.846%**	3.738%
	Occupied	0.144%	0.138%	**89.828%**
Our Method (Key Frames)	Unknown	**99.696%**	0.076%	11.694%
	Free	0.062%	**99.848%**	2.806%
	Occupied	0.242%	0.076%	**85.500%**
Cartographer	Unknown	96.868%	0.593%	23.943%
	Free	1.743%	98.584%	50.795%
	Occupied	1.389%	0.823%	25.262%

TABLE 3.5: OGM Precision of Our Method Using All Frames, Our Method Using Key Frames, and Cartographer – Simulation 3.

		Ground Truth		
		Unknown	Free	Occupied
Predicted				
Our Method (All Frames)	Unknown	**99.812%**	0.032%	4.500%
	Free	0.036%	**99.928%**	2.358%
	Occupied	0.156%	0.036%	**93.142%**
Our Method (Key Frames)	Unknown	**99.258%**	0.554%	16.788%
	Free	0.430%	**99.352%**	3.736%
	Occupied	0.312%	0.094%	**79.476%**
Cartographer	Unknown	96.968%	1.018%	26.420%
	Free	1.574%	98.110%	44.928%
	Occupied	1.458%	0.872%	28.652%

uncorrected. Although global pose graph optimization is applied at the end of the process, it often suffers from an excess of inaccurate and conflicting relative measurements, as well as its susceptibility to local minima, limiting its effectiveness in correcting these errors. Moreover, pose graph optimization typically does not enhance the local details of maps, as it focuses solely on optimizing poses without jointly considering the map. This further highlights the advantage of our approach, which jointly optimizes both robot poses and the occupancy map.

The mapping performance of our method and Cartographer using 3 simulated datasets is summarized in Table 3.3, Table 3.4, and Table 3.5, clearly showing that both variants of

our method, one using all frames and the other using keyframes, significantly outperform Cartographer in terms of map accuracy.

TABLE 3.6: Accuracy of the OGM.

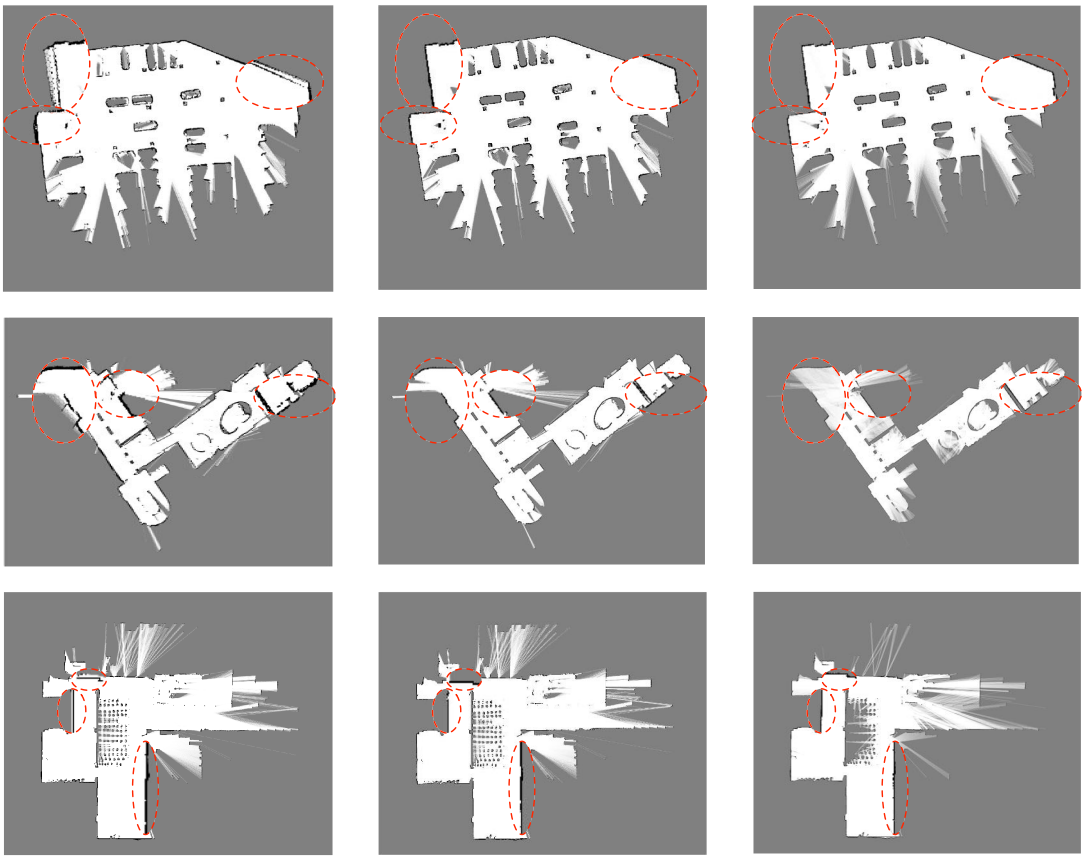
		AUC	Precision
Simulation 1	Cartographer	0.90878	0.95651
	Ours (All)	**0.99999**	**0.99773**
	Ours (Key)	**0.99902**	**0.99548**
Simulation 2	Cartographer	0.96132	0.96829
	Ours (All)	**0.99926**	**0.99721**
	Ours (Key)	**0.99914**	**0.99638**
Simulation 3	Cartographer	0.92696	0.96592
	Ours (All)	**0.99974**	**0.99771**
	Ours (Key)	**0.99748**	**0.99113**

3.4.2 Comparisons Using Practical Datasets

We use three normal-scale practical datasets, namely Deutsches Museum b0 [8], Car Park [94], and C5, to compare our method with Cartographer in terms of the constructed OGMs and optimized poses.

The mapping quality is evaluated by comparing the details of the constructed maps. Additionally, point cloud maps, which are generated using the endpoint projections of scan points and optimized poses, serve as a reference for pose accuracy. For the Car Park and C5 datasets, our method is initialized with poses from odometry inputs, whereas for the Museum b0 dataset, initialization relies on poses from scan matching due to the absence of odometry. The OGMs and point cloud maps generated by Cartographer, our method using all frames, and our method using key frames for the three datasets are shown in Figure 3.9 and Figure 3.10. Red dotted lines highlight areas where our results outperform Cartographer in both the OGMs and point cloud maps. Comparing Figure 3.9(a) and Figure 3.9(b), our method provides more precise boundaries for the OGMs due to joint optimization of robot poses and the occupancy map. Similarly, the comparison between Figure 3.10(a) and Figure 3.10(b) illustrates that our method achieves more accurate poses.

Moreover, our method outperforms Cartographer when using only key frames, as evident from the comparison of Figure 3.9(a) and Figure 3.10(a) with Figure 3.9(c) and Figure



(a) Cartographer (b) Ours (All) (c) Ours (Key)

FIGURE 3.9: **The OGMs from Cartographer, our method using all frames, and our method using key frames.**

TABLE 3.7: Time Consumption of Different Algorithms.

Dataset	Computational Time (s)		
	Cartographer	Ours (All)	Ours (Key)
Car Park	168	119	**44**
Museum b0	152	126	**38**
C5	146	137	**35**

3.10(c). These results show that, despite Cartographer introducing loop closure detection, it still produces non-negligible pose errors, leading to point clouds that fail to fully overlap observations of the same obstacle at different poses. While the point cloud maps generated by our method also have non-overlapping parts, these areas are significantly smaller compared to those from Cartographer.

Additionally, we assess the time consumption of our method and Cartographer on these three datasets. Table 3.7 shows that our method consistently requires less time than

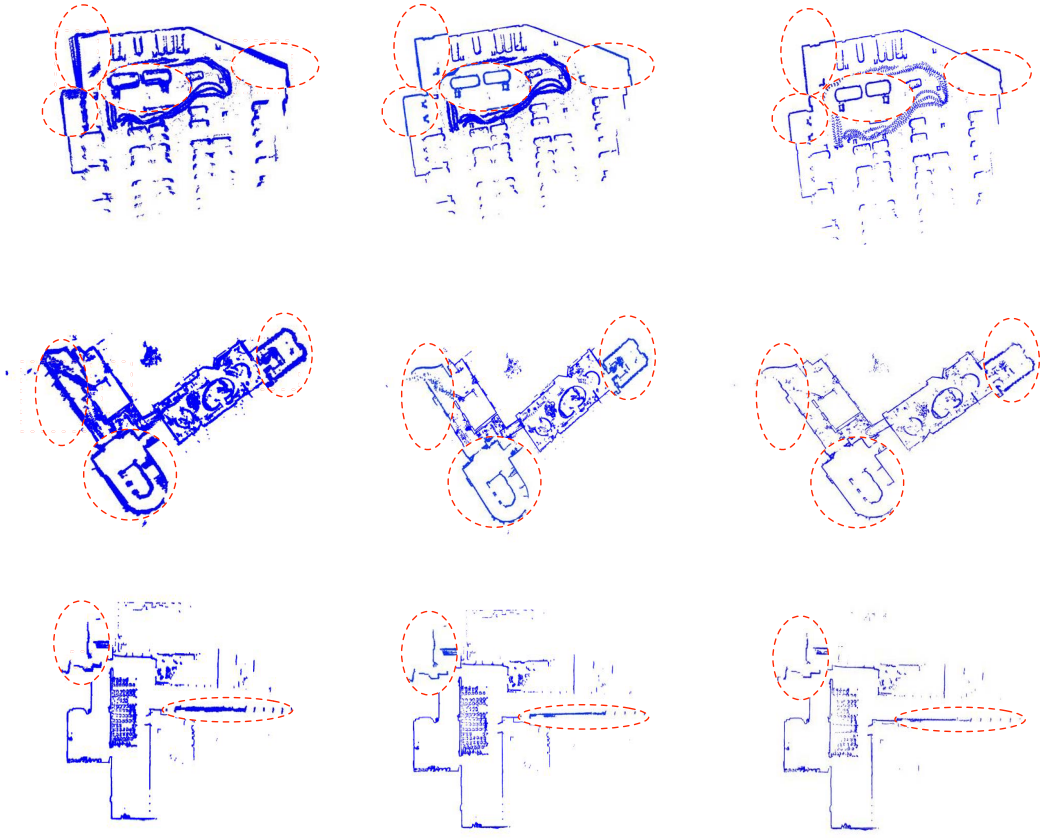


FIGURE 3.10: **The point cloud maps from Cartographer, our method using all frames, and our method using key frames**.

Cartographer across all datasets when using all frames and achieves significantly better efficiency when using selected key frames.

Finally, it is worth noting that some well-known public datasets, such as Radish [125], were collected before 2014 with outdated sensors, leading to low-quality data with poor scanning frequency and odometry accuracy. These issues hinder the performance of Cartographer, often requiring meticulous parameter tuning but still yielding suboptimal results. In contrast, our method performs well on these datasets. Although we do not include these comparisons in this thesis, we make our results available on our code page[1].

[1] `https://github.com/WANGYINGYU/Occupancy-SLAM`

TABLE 3.8: Robustness to Initialization.

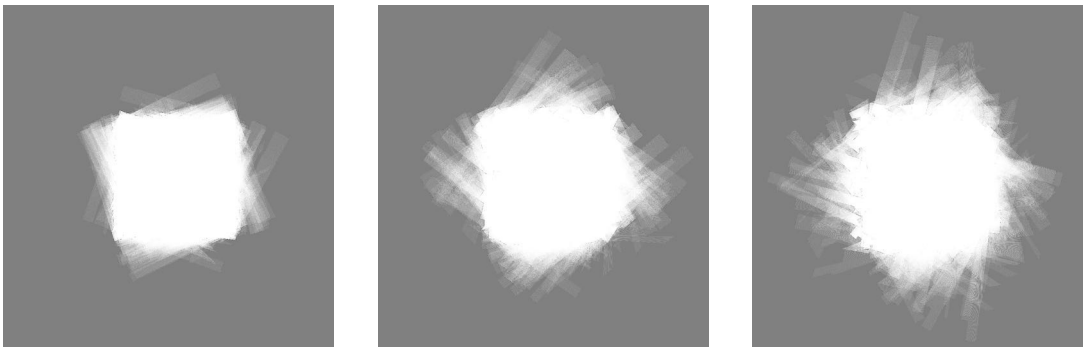
Noise Level	Convergence Percentage	Average MAE of Translation (m)	Average MAE of Rotation (rad)
Level 1 (2 m, 0.5 rad)	100%	0.00679	0.0005
Level 2 (4 m, 1 rad)	100%	0.00682	0.0005
Level 3 (6 m, 1.5 rad)	80%	0.01742	0.0012

3.4.3 Assessment of Robustness to Initial Guess

While our method has demonstrated robustness when initialized with odometry inputs or scan matching under reliable sensor conditions in both simulation and real-world experiments, this subsection highlights its capability to converge even when initialized with significantly noisy poses. We use all frames in this subsection to assess robustness.

First, we use Simulation 1 dataset to quantitatively evaluate the convergence percentage and the accuracy of optimized poses under different noise levels. We add zero-mean uniformly distributed noises with different bounds to the ground truth of the poses to generate each group of ten sets of initial poses for the experiments to count convergence rates and average errors. Specifically, for noise level 1, the noise for translation is within [−2 m, 2 m] and the noise for rotation is within [−0.5 rad, 0.5 rad]; for level 2, [−4 m, 4 m] and [−1 rad, 1 rad]; for level 3, [−6 m, 6 m] and [−1.5 rad, 1.5 rad]. The poses with different noise levels of Simulation 1 dataset are visualized using the generated OGMs, as shown in Figure 3.11. The convergence results are depicted in Table 3.8, showing that our method can 100% converge when initialized with challenging noisy poses of level 1 and level 2. Our method still has a high convergence percentage (80%) when initialized with noisy poses of level 3. Our algorithm using other simulation datasets has a similar robustness performance.

Moreover, for all practical datasets, we additionally add random zero-mean uniform distribution noises ([−2 m, 2 m] for translation and [−0.5 rad, 0.5 rad] for rotation) to the poses obtained from Cartographer as the initial guess. The initial occupancy maps obtained by using the noisy initial poses are shown in Figure 3.12(a). Figure 3.12(b) shows the remapped OGMs using our optimized poses, and Figure 3.12(c) shows the point cloud



(a) Level 1 (b) Level 2 (c) Level 3

FIGURE 3.11: **Examples of OGMs generated from poses with different noise levels as shown in Table 3.8 using Simulation 1 dataset.**

maps using our optimized poses. This experiment shows that our approach can converge from initial guesses with significant errors and also generate good results.

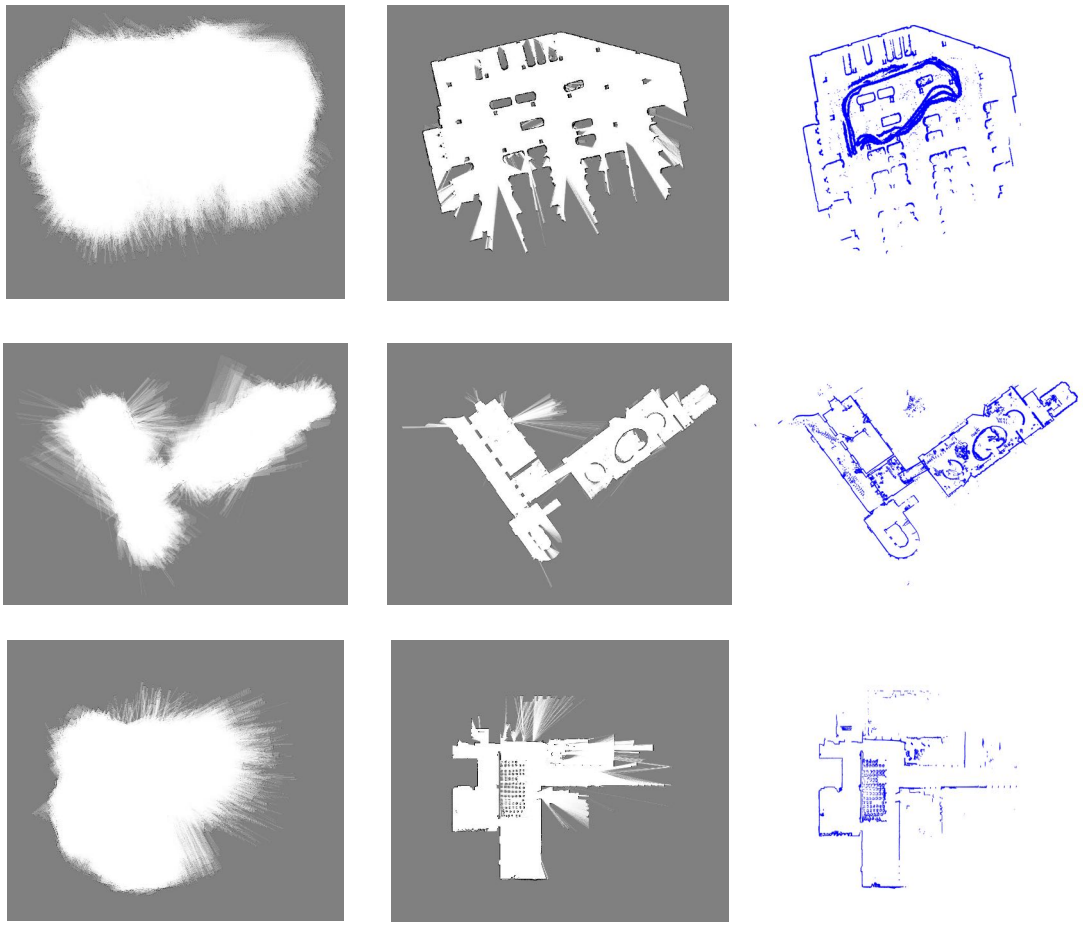
3.4.4 Discussion about the Effectiveness of Different Stages

In previous sections, we demonstrated the accuracy, robustness, and efficiency of our proposed method. In this section, we discuss the effectiveness of its different parts.

As demonstrated in Table 3.2, Figure 3.8, Figure 3.9, and Figure 3.10, the accuracy of the poses and the map obtained from our full approach (Algorithm 4) is much better than those obtained from Cartographer. This confirms the advantage of jointly optimizing both the robot poses and the occupancy map.

One potential question is whether using only Algorithm 1 with a high-resolution map would yield even better results. To investigate this, we compared our full approach with Algorithm 1 using a high-resolution map. We tested three initialization: (1) *Algorithm 1 (High, O/S)*: initialization using odometry inputs or scan matching; (2) *Algorithm 1 (High, Carto)*: initialization using Cartographer's poses (as proposed in our conference paper [15]); and (3) *Algorithm 1 (High, First)*: initialization using the poses obtained by our first stage.

First, *Algorithm 1 (High, O/S)* fails to converge on most datasets, while our full method converges successfully, indicating the improved robustness of our multi-resolution strategy.

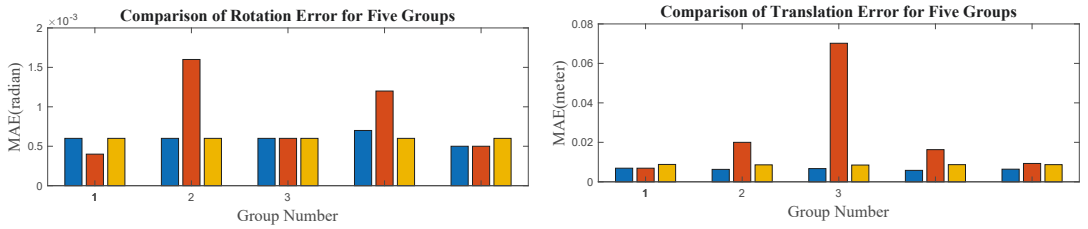


(a) OGM Mapped by Initial Poses (b) Our Remapped OGM (c) Our Point Cloud

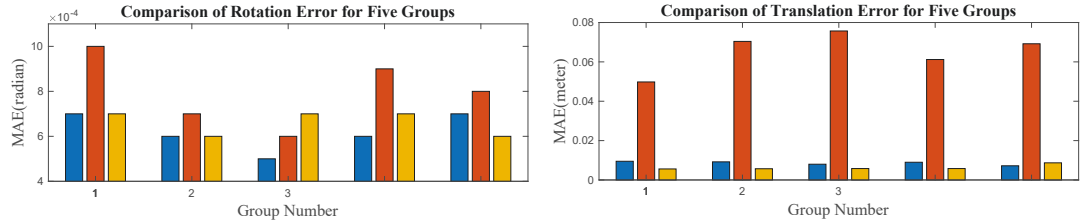
FIGURE 3.12: **The OGMs and point cloud maps generated using noisy poses for initialization by our approach.** (a) and (b) display the remapped occupancy maps generated from the noisy initial poses and our optimized poses, respectively, and (c) shows the point cloud maps created by projecting the endpoints of scans using our optimized poses.

The comparison between our full method, *Algorithm 1 (High, Carto)*, and *Algorithm 1 (High, First)* across all five simulation groups is shown in Figure 3.13. It can be observed that the accuracy of our full method remains stable across all groups, while the accuracy of *Algorithm 1 (High, Carto)* varies drastically. This suggests that the approach in our conference paper [15] not only requires an accurate initial guess but also produces less accurate poses than our new method. Moreover, comparing *Algorithm 1 (High, Carto)* with *Algorithm 1 (High, First)* further confirms that the poses obtained in our first stage are more accurate than those of Cartographer.

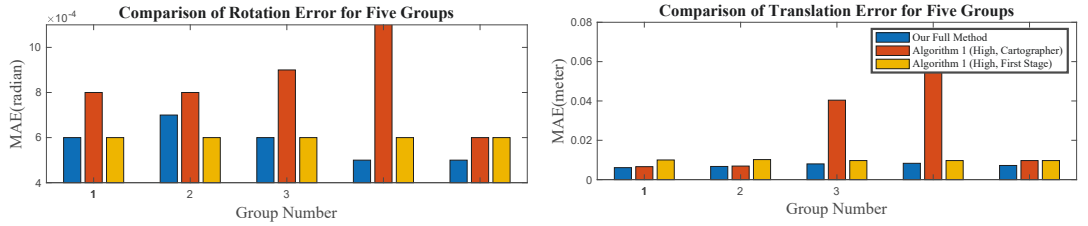
It is also worth noting that our full method utilizes the selected high-resolution map for



(a) Simulation 1



(b) Simulation 2



(c) Simulation 3

FIGURE 3.13: **Comparison of rotation and translation errors for simulated datasets using three methods:** our full method (Algorithm 4); Algorithm 1 initialized with Cartographer poses and a high-resolution map [15]; and Algorithm 1 initialized with poses from our first stage and a high-resolution map.

optimization in the second stage. As shown in Figure 3.13, in certain experiments, the accuracy of our full approach surpasses that of the optimization using the full high-resolution map (*Algorithm 1 (High, First)*). A possible explanation is that once relatively accurate poses and occupancy maps are obtained, the dropped cell vertices and corresponding observations contain little information. Retaining all cells and corresponding observations for optimization may prevent the algorithm from finding the optimal solution, as all are observation items given uniform weights. Another reason could be the smoothing term in (3.10), which spreads occupancy values to unknown cell vertices, potentially introducing errors that affect the convergence of the optimization.

In terms of time consumption, our full approach is much more efficient than *Algorithm 1 (High, Carto)*. For instance, *Algorithm 1 (High, Carto)* consumes over 21,000 seconds

TABLE 3.9: Impact of First-Stage Resolution Settings.

		$r = 20$	$r = 10$	$r = 5$	$r = 2$
Simulation 1	MAE/Trans (m) First	0.02352	0.02206	**0.02118**	0.17318
	MAE/Rot (rad) First	0.00116	**0.00098**	0.00124	0.01066
	MAE/Trans (m) All	0.00812	**0.00640**	0.00728	0.16066
	MAE/Rot (rad) All	0.00062	0.00060	**0.00054**	0.01008
	Total Time (s)	**118**	148	262	2183
Simulation 2	MAE/Trans (m) First	0.03938	0.03224	**0.01984**	0.09160
	MAE/Rot (rad) First	0.00332	0.00220	**0.00108**	0.00314
	MAE/Trans (m) All	0.01742	0.00858	**0.00584**	0.08018
	MAE/Rot (rad) All	0.00064	0.00062	**0.00052**	0.00286
	Total Time (s)	**149**	193	321	2685
Simulation 3	MAE/Trans (m) First	0.06708	0.02624	**0.01776**	0.03570
	MAE/Rot (rad) First	0.00384	0.00164	**0.00124**	0.00278
	MAE/Trans (m) All	0.01586	**0.00726**	0.00816	0.02082
	MAE/Rot (rad) All	0.00100	**0.00058**	0.00068	0.00102
	Total Time (s)	**125**	132	185	1041

with the Car Park dataset. In comparison, the time consumption of our full approach using all frames is 119 seconds (less than 0.6%), and using only key frames, it takes only 44 seconds (about 0.2%). This substantial reduction in time consumption highlights the efficiency improvements of our method over our conference paper [15].

The reduction in time consumption stems from both the multi-resolution strategy, which reduces time per iteration, and the fewer iterations needed in the second stage due to the selected high-resolution map. Our experiments show that only about two iterations are needed in the second stage with the selected high-resolution map, fewer than in *Algorithm 1 (High, First)*. This is likely because the selected high-resolution map focuses on critical states, with observations containing the most relevant information, enabling faster convergence.

In summary, compared to our conference paper [15], our new multi-resolution method does not require precise initialization, is far more efficient, and achieves higher accuracy.

3.4.5 Ablation Study on the Resolution Ratio

In this section, we perform ablation experiments on simulation datasets to analyze the impact of varying resolution settings in the first stage of the multi-resolution strategy on

overall optimization performance.

We assess accuracy and computational time using three simulation datasets, with the resolution in the second stage fixed at $s^h = 0.05$ m. The resolution ratios r between the first and second stages are set to 2, 5, 10, and 20, respectively. To ensure consistency, a fixed selection range of $d = 1.5$ m is applied uniformly across all datasets.

The results, shown in Table 3.9, reveal that $r = 10$ achieves the best trade-off between time consumption and accuracy. While $r = 20$ minimizes time consumption, it reduces the accuracy of poses in the first stage, adversely impacting final optimization accuracy. Conversely, $r = 5$ improves pose accuracy in the first stage at the cost of higher time consumption but does not consistently enhance final accuracy. Notably, r may need adjustment for other high-resolution settings.

3.4.6 Computational Complexity Analysis

In this section, we analyze the computational complexity and evaluate the time consumption of our method using large-scale datasets.

The GN method for solving the joint optimization of occupancy maps and poses in (3.17) primarily depends on calculating Jacobian $\mathbf{J}$, and constructing and solving the sparse linear system in (3.18) [101].

The objective function consists of the observation term, the odometry term, and the smoothing term. Let $\lambda(\mathbb{S})$ denotes the number of sampling points $\mathbb{S}$, then the number of items in the objective function is $\mathfrak{d}_{row} = \lambda(\mathbb{S}) + 3(n-1) + 2c_w c_h + c_w + c_h$, and the state vector size is $\mathfrak{d}_{col} = 3n + (c_w + 1)(c_h + 1)$. Considering Jacobian of the smoothing term $\mathbf{J}_S$ can be pre-calculated before optimization, the number of non-zero elements of Jacobian matrix that need to be computed for each iteration is $\mathfrak{d}_J = 7\lambda(\mathbb{S}) + 6(n-1)$. Therefore, for each iteration, the computation complexity of Jacobian calculation, constructing (3.18) and solving (3.18) is $\mathcal{O}(\mathfrak{d}_J)$, $\mathcal{O}(\mathfrak{d}_J \mathfrak{d}_{col})$, and $\mathcal{O}(\mathfrak{d}_{col}^3)$, respectively. Therefore, the total computation complexity per iteration for the local map and poses joint optimization problem is $\mathcal{O}(\mathfrak{d}_J + \mathfrak{d}_J \mathfrak{d}_{col} + \mathfrak{d}_{col}^3)$. Due to our proposed multi-resolution joint optimization strategy

and keyframe selection, both $\mathfrak{d}_J$ and $\mathfrak{d}_{col}$ remain small during the first and second stages of optimization, making the computation time for this part manageable.

3.5 Extension of the Algorithms to 3D Case

In this section, we extend our approach for jointly optimizing robot poses and the occupancy map from 2D to 3D environments, and present experimental results demonstrating the superior performance of our method compared to state-of-the-art approaches.

3.5.1 3D Adaptation

Our approach for jointly optimizing robot poses and occupancy maps extends naturally to 3D, where observations, robot trajectories, and map representations are all defined in 3D space. Most of the original problem formulations and algorithmic components can be adapted to the 3D case with only minor modifications.

In the 3D formulation, observations transition from 2D laser scans to 3D LiDAR point clouds, while robot poses and odometry are expressed in six degrees of freedom (DoF). The occupancy map is represented as a 3D volumetric grid. As a result, the interpolation operations in equations (3.1) and (3.9) are updated from bilinear to trilinear interpolation, along with their corresponding inverse operations.

For the objective function (3.10), the odometry term (3.14) should be replaced with a 6 DoF odometry term for 3D, and the smoothing term (3.15) should include a smoothing penalty for the z-axis, Jacobians $\mathbf{J}_P$, $\mathbf{J}_M$, $\mathbf{J}_O$, and $\mathbf{J}_S$ described in Appendices need to be adjusted accordingly.

3.5.2 3D Experimental Results

3.5.2.1 Evaluation Metrics and State-of-the-art Methods

We evaluate our method's performance in 3D using absolute trajectory error (ATE) for poses, aligning and comparing results with ground truth via EVO [126], as used in [78,

116]. In all the experiments, we use the odometry information provided by the dataset as initialization if it is available. Otherwise, we use FAST-LIO2 [127] to obtain the odometry information. To evaluate our method, we compare our method against state-of-the-art methods: FAST-LIO2 [127] and BALM2 [115]. BALM2 optimizes the planar feature parameters of the point cloud and the robot's poses.

3.5.2.2 Datasets

We perform comparisons using two real-world datasets. (1) The Newer College Dataset [128]: The first five sequences from the *shorter experiment*, collected with a handheld Ouster OS-1 LiDAR scanner at New College, Oxford. The environment includes lawns, buildings, a tunnel, and a garden. Ground truth is provided by a BLK360 LiDAR scanner to capture a detailed 3D map and then infer the ground truth of poses with centimeter-level accuracy. (2) Arche Dataset [10]: A demo dataset for Voxgraph, collected using an Ouster OS1 LiDAR mounted on a hexacopter MAV in a disaster area. Ground truth positions are provided by an RTK-GNSS system.

The Newer College Dataset is used to evaluate high-precision performance in normal-scale environments, while the KITTI and Arche datasets are used to test performance in large environments with long trajectories.

3.5.2.3 Experimental Results

We evaluate the performance of our proposed method without submap joining in normal-scale environments.

First, we evaluate BALM2 and our method using the first five sequences of The Newer College Dataset, which encompass all scenarios within the dataset. As shown in Table 3.10, our method outperforms BALM2 across all metrics, except for the RMSE in Seq. 1, and significantly outperforms the odometry inputs from FAST-LIO2 in all metrics.

Next, we test robustness in a challenging environment with noisy odometry input using the Arche dataset. This dataset, collected by a hexacopter MAV in an unstructured

TABLE 3.10: Absolute Trajectory Error (MAE/RMSE, Meters) in Normal-scale Environments for Different 3D Methods.

Method	Seq. 0	Seq. 1	Seq. 2	Seq. 3	Seq. 4
FAST-LIO2	0.518/0.717	0.181/0.202	0.121/0.132	0.188/0.200	0.571/0.723
BALM2	0.283/0.326	0.112/**0.123**	0.104/0.109	0.144/0.158	0.298/0.344
Ours	**0.185/0.232**	**0.097/0.123**	**0.091/0.099**	**0.141/0.155**	**0.238/0.284**

environment, is influenced by drone vibrations, flight speed, and environmental factors. Local point cloud maps built using odometry from ROVIO [129] (also used to construct submaps in Voxgraph) are shown in the first row of Figure 3.14. To evaluate BALM2 and our method, we partition the dataset into several short sequences, each lasting 10–20 seconds. BALM2 fails in all the sequences except during MAV start-up and landing due to insufficient planar features for optimization, while our method performs well on all the sequences. The second row of Figure 3.14 illustrates some of our results, demonstrating that our method is robust in 3D and does not rely on environmental assumptions. Additionally, the results confirm that our method achieves significantly higher pose accuracy than ROVIO.

3.5.3 Discussion

The experimental results in this section demonstrate that our proposed idea of jointly optimizing the robot pose and the occupancy map can also lead to better solutions for the robot poses and occupancy maps in 3D cases. However, several challenges remain in 3D scenarios. For instance, 3D point clouds from LiDAR scanners are often sparse, particularly in the vertical direction, which can lead to observability issues in the optimization problem. This sparsity also results in inhomogeneous observation information, complicating the accurate representation of the 3D environment in occupancy maps. Furthermore, the large dimensions of 3D maps pose significant computational challenges in large-scale SLAM, requiring more efficient solving methods.

To address these challenges, several potential solutions can be explored. First, adopting compact representations for 3D occupancy maps, such as octree structures similar to Octomap [22] and [77], can enhance efficiency. Second, combining local map and pose optimization with hierarchical optimization and submap joining methods can further reduce

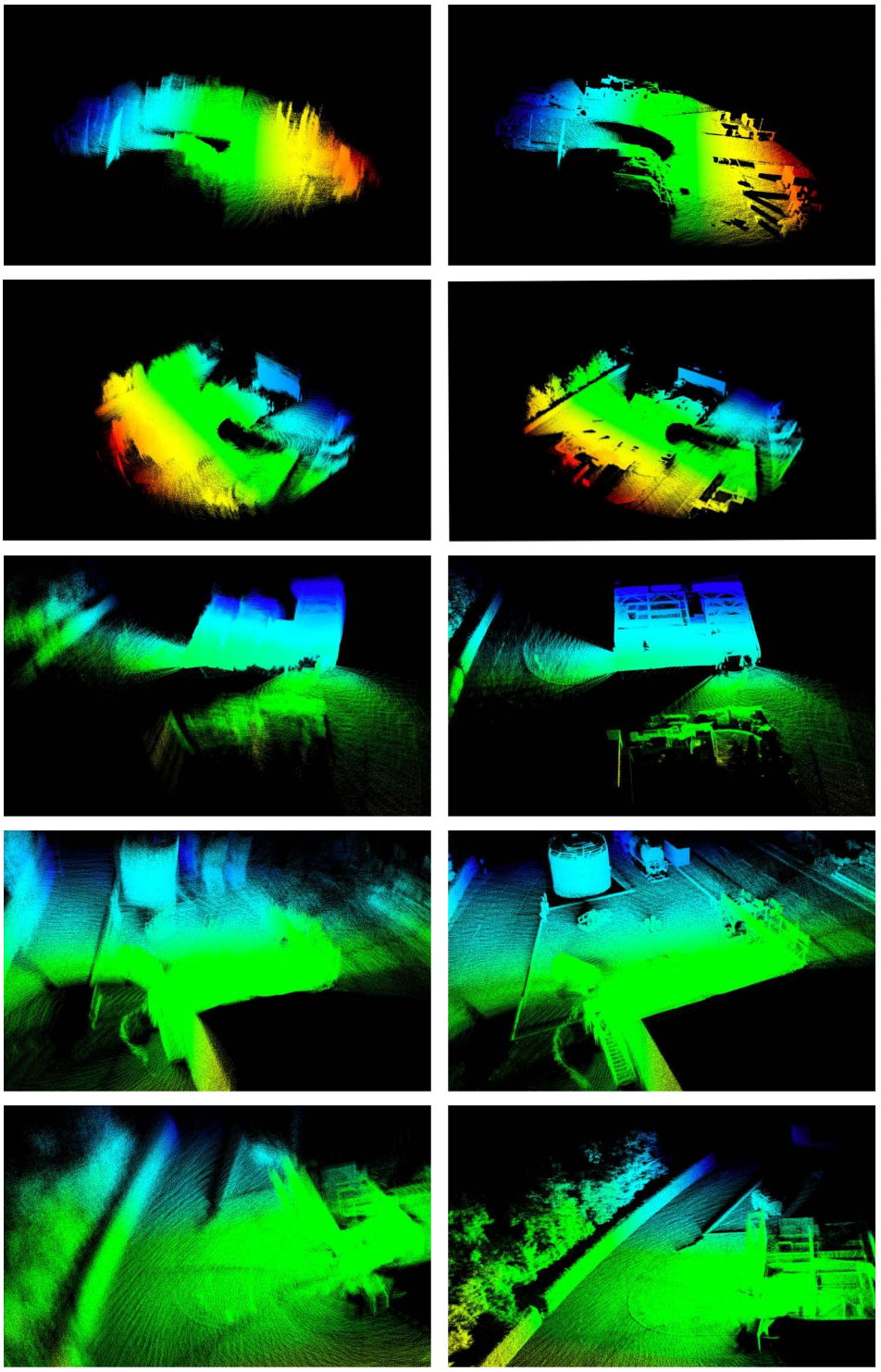


FIGURE 3.14: **Some local point cloud maps from the Arche dataset.** The first column shows point cloud maps generated using odometry from ROVIO (also used for submap construction in Voxgraph), while the second column shows maps generated with our optimized poses using the same LiDAR scans. BALM2 fails to produce results when using the same odometry and scans as inputs in all these local environments.

computational time. Lastly, using continuous representations for 3D occupancy maps enables more precise gradient calculations, which can better guide the optimization process.

3.6 Chapter Summary

In this chapter, we propose Occupancy-SLAM algorithm, which solves robot poses and occupancy map simultaneously. To enhance efficiency and robustness, we introduce a multi-resolution strategy. The first stage jointly optimizes poses and a low-resolution occupancy map to quickly achieve relatively accurate pose estimates, which are then used as the initial guess for the second stage. The second stage refines poses and a subset of the high-resolution map, focusing on critical boundary areas. Results from both simulated and real-world datasets demonstrate that our method achieves more accurate pose and map estimates than state-of-the-art approaches.

Our findings show that solving poses and occupancy maps simultaneously yields more accurate results compared to first solving pose-graph SLAM and then constructing the map. This joint optimization approach has the potential to revolutionize occupancy map based SLAM frameworks.

The proposed method acts as a batch optimization approach for obtaining high-quality robot poses and maps. Unlike incremental or online methods, batch optimization provides greater accuracy, which is particularly advantageous for applications requiring high-quality maps rather than real-time operation (e.g., offline map creation for precise future localization). Despite typical drawbacks of batch optimization, such as higher computational costs, trajectory-length-dependent complexity, and reliance on accurate initial guesses, our method effectively overcomes these limitations: 1) our method is efficient due to the proposed multi-resolution joint optimization strategy, and the computation time is comparable to online methods; 2) our method can use selected keyframes to further reduce the computational cost without losing too much accuracy; and 3) our method is very robust to the initial guess and can be initialized from odometry inputs or scan matching, so it does not require initialization from the result of incremental/online methods.

Although the proposed method in this chapter effectively solves the SLAM problem in normal-scale environments, as the robot continues to operate, the trajectory becomes increasingly long, which significantly increases the complexity of the optimization problem. Therefore, it is necessary to develop an algorithm that can address the challenges of large-scale SLAM.

Chapter 4

Joint Optimization of Global Occupancy Map and Local Submap Frames

In Chapter 3, we introduced a method for the simultaneous optimization of robot poses and the occupancy map, and experimental results demonstrate that this approach can effectively improve SLAM accuracy. However, as the robot trajectory becomes increasingly long, the computational complexity grows accordingly. This scalability limitation hinders the applicability of the Occupancy-SLAM algorithm in large-scale environments and long-duration trajectories.

To address this issue, in this chapter, we propose a grid-based submap joining algorithm that enables joint optimization of the global occupancy map and local submap frames. This approach significantly reduces computational demands and allows the method to scale to larger environments. Moreover, the submap joining strategy enhances the robustness of Occupancy-SLAM by mitigating the risk of becoming trapped in local minima during optimization in large-scale scenarios.

Specifically, we first formulate the grid-based submap joining problem as a NLLS form to optimize the global occupancy map and local submap frames simultaneously. We then prove that in solving the NLLS problem using GN method, the increments of the poses

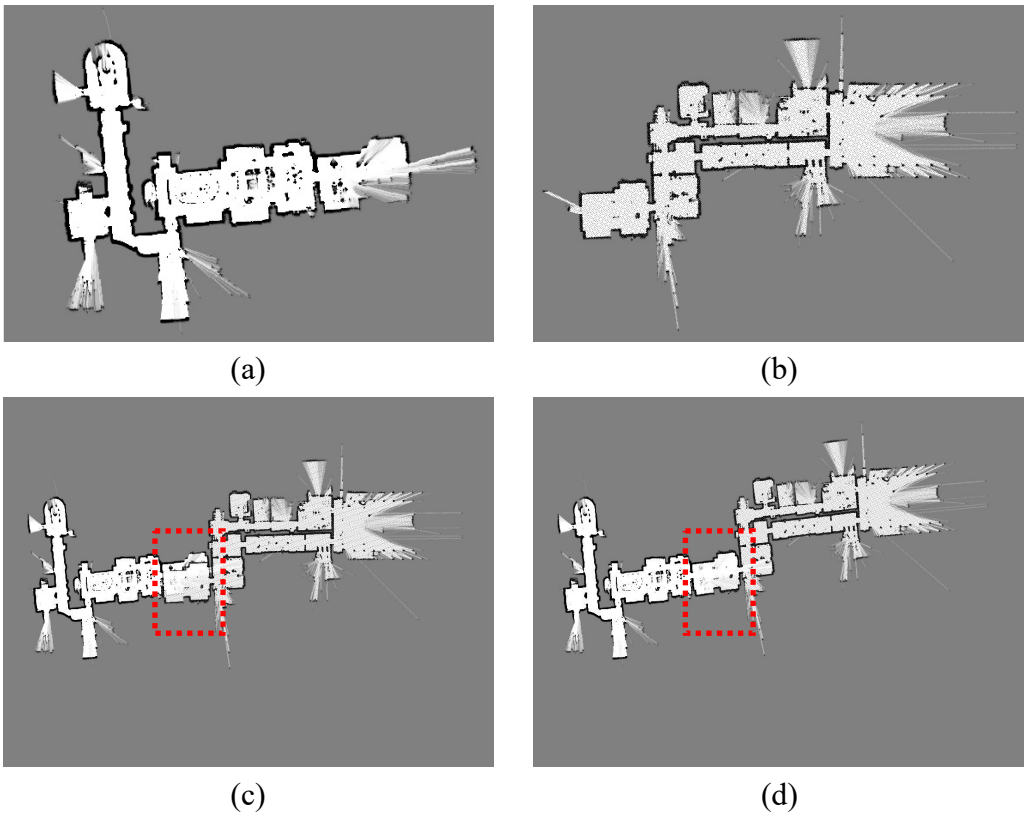


FIGURE 4.1: **The description of the grid-based submap joining problem.** (a) and (b) show the grid-based submaps. (c) shows the global OGM generated by fusing the two submaps using the pose information only. (d) shows the consistent global OGM generated by the proposed grid-based submap joining method.

in each iteration are independent of the occupancy values of the global occupancy map. Based on this property, we propose a pose-only GN algorithm equivalent to full GN method to solve the NLLS problem. The proposed submap joining algorithm is very efficient due to the independent property and the pose-only solution.

In this chapter, we first present the formulation in the 2D case and then extend this approach to the 3D scenario.

4.1 Problem Formulation: Grid-based Submap Joining

In this section, different from [8, 130], we formulate the grid-based submap joining problem as a NLLS problem in which the variables are not the robot poses, but the global grid

map and the local submap frames. A description of the local grid-based submap joining problem is shown in Figure 4.1.

4.1.1 Inputs and Outputs of Submap Joining Problem

For the local submap joining problem, the input is a sequence of grid-based local submaps. First, let us denote $n+1$ submaps as $\mathbb{L} = \{\mathbb{L}_0, \cdots, \mathbb{L}_n\}$, where $\mathbb{L}_i$ represents the i-th local occupancy map built by any evidence grid mapping technique such as [8, 15, 20]. Each submap $\mathbb{L}_i$ stores the occupancy values at all discrete cells in this submap, expressed as the log-odds ratio — the logarithm of the ratio between the probability of a cell being occupied and the probability of it being free. Different from [8], we do not maintain all the robot poses in each submap. The only poses we care about are the submap frames. Assume that the coordinate frames of these local submaps in the global map coordinate frame are $\{\mathbf{x}_0^P, \ldots, \mathbf{x}_n^P\} \in \mathbb{SE}(2)^{n+1}$, where $\mathbf{x}_i^P = [\mathbf{t}_i^\top, \theta_i]^\top$ represents the i-th local submap coordinate frame. $\mathbf{t}_i$ is the x-y position and θ_i is the orientation with the corresponding rotation matrix $\mathbf{R}_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}$.

The task of the submap joining problem is to find the optimal poses of local submap coordinate frames and the optimal global grid map. We denote the global grid map as $\mathbb{M} = \{M(\mathbf{m}_{00}), \cdots, M(\mathbf{m}_{c_w c_h})\}$, where $\mathbf{m}_{wh}$ $(0 \leq w \leq c_w, 0 \leq h \leq c_h)$ represents the coordinate of a discrete cell in the global map and $M(\mathbf{m}_{wh})$ represents its corresponding occupancy value [121]. Therefore, the outputs of submap joining problem are the optimal solution of local submap coordinate frames and the optimal global map.

4.1.2 Global to Local Coordinate Projection

The grid cell $\mathbf{m}_{wh}$ in the global map $\mathbb{M}$ can be projected to $i-$th local submap coordinate by pose $\mathbf{x}_i^P$, i.e.,

$$\mathbf{p}_i^{wh} = \frac{\mathbf{R}_i^\top (\mathbf{m}_{wh} \cdot s_G - \mathbf{t}_i)}{s_L}, \tag{4.1}$$

where $\mathbf{p}_i^{wh}$ is the projected position in the local submap coordinate, s_G and s_L are the resolutions of cells in the global occupancy map and local submaps, respectively.

4.1.3 NLLS Formulation

The state vector of the proposed map joining problem is

$$\mathbf{x} = \left[\mathbf{x}^{P\top}, \mathbf{x}^{M\top}\right]^\top, \tag{4.2}$$

where

$$\begin{aligned} \mathbf{x}^P &= \left[\left(\mathbf{x}_1^P\right)^\top, \cdots, \left(\mathbf{x}_n^P\right)^\top\right]^\top \\ \mathbf{x}^M &= \left[M\left(\mathbf{m}_{00}\right), \cdots, M\left(\mathbf{m}_{c_w c_h}\right)\right]^\top. \end{aligned} \tag{4.3}$$

As with most submap joining problem formulations, we fix the first local map coordinate frame as the global coordinate. Therefore, $\mathbf{x}^P$ consists of n local map coordinate frames and $\mathbf{x}^M$ includes $(c_w + 1) \times (c_h + 1)$ discrete cells of global occupancy map.

By the global-to-local projection relationship, all cells of global occupancy map $\mathbb{M}$ can be projected to corresponding submaps to compute the difference in occupancy values to formulate the NLLS problem, i.e., minimize

$$f(\mathbf{x}) = \sum_{i=0}^{n} \sum_{wh \in \mathbb{S}_i} \left\| \omega(i, \mathbf{m}_{wh}) M(\mathbf{m}_{wh}) - L_i(\mathbf{p}_i^{wh}) \right\|^2 \tag{4.4}$$

where $\mathbb{S}_i$ is the set including the indices of cells in the global occupancy map $\mathbb{M}$ projected onto local submap $\mathbb{L}_i$. $M(\mathbf{m}_{wh})$ refers to the occupancy value of the cell $\mathbf{m}_{wh}$ in the global occupancy map $\mathbb{M}$. $L_i(\mathbf{p}_i^{wh})$ means the occupancy value of the continuous coordinate $\mathbf{p}_i^{wh}$ in the local submap $\mathbb{L}_i$. Here we use bilinear interpolation to obtain the occupancy value $L_i(\mathbf{p}_i^{wh})$ on the discrete grid-based submap $\mathbb{L}_i$, similar to [14, 15].

In (4.4), $\omega(i, \mathbf{m}_{wh})$ is a function to evaluate the weight to establish an accurate relationship between the global map and local submaps w.r.t. occupancy values of corresponding coordinates because both local submaps and the global map are represented by the Bayesian approach [21]. It can be calculated by

$$\omega(i, \mathbf{m}_{wh}) = \frac{N^{L_i}(\mathbf{p}_i^{wh})}{N^M(\mathbf{m}_{wh})}. \tag{4.5}$$

Here, N^{L_i} is a function to evaluate the hit number at arbitrary points on submap $\mathbb{L}_i$, which is similar to [15, 16]. $N^{L_i}(\mathbf{p}_i^{wh})$ approximates the observation count of the continuous coordinate $\mathbf{p}_i^{wh}$ using bilinear interpolation based on the observation count of surrounding discrete cells. N^M denotes the global hit map associated with $\mathbb{M}$ which is built by projecting all the local hit maps $\mathbb{N}^L = \{N^{L_i}\}$ into the global map coordinate frame through $\mathbf{x}^P$. Thus, $\omega(i, m_{wh})$ is a function of $\mathbf{x}^P$.

4.2 Efficient Pose-only GN Algorithm

In this section, we prove the existence of a special independent property when solving the proposed NLLS formulation (4.4) using GN method. Based on this special property, we propose a pose-only GN algorithm equivalent to full GN method to solve the proposed NLLS problem.

4.2.1 Iterative Solver to the NLLS Formulation

The proposed submap joining problem is to seek $\mathbf{x}$ to minimize $f(\mathbf{x})$ in (4.4). Typical iterative algorithms, such as GN method, can be used to solve (4.4) iteratively, i.e., by starting with an initial guess $\mathbf{x}(0)$ and updating with $\mathbf{x}(k+1) = \mathbf{x}(k) + \boldsymbol{\Delta}(k)$. The update vector $\boldsymbol{\Delta}(k) = \left[\boldsymbol{\Delta}^P(k)^\top, \boldsymbol{\Delta}^M(k)\right]^\top$ is the solution to

$$\mathbf{J}^\top \mathbf{J} \boldsymbol{\Delta}(k) = -\mathbf{J}^\top F(\mathbf{x}(k)) \tag{4.6}$$

where $F(\mathbf{x})^\top F(\mathbf{x}) = f(\mathbf{x})$ and $\mathbf{J}$ is the Jacobian matrix $\partial F / \partial \mathbf{x}$ evaluated at $\mathbf{x}(k)$. However, for grid-based submap joining problem, the number of cells $(c_w+1)\times(c_h+1)$ in the global map $\mathbb{M}$ is very large which leads to a high dimension of the state vector, so common methods to solve (4.6) are time-consuming.

4.2.2 A Special Independent Property of the Proposed Formulation

The proposed NLLS formulation (4.4) can be shown to have a special property, when optimizing poses and the global occupancy map together using GN method, the optimization of poses is independent of the global occupancy map.

Proposition. *For each step of GN iteration for minimizing the NLLS problem in (4.4), the increment of poses $\boldsymbol{\Delta}^P(k)$ is independent of the occupancy values of the global occupancy map $\mathbf{x}^M$.*

Proof. The Jacobian matrix $\mathbf{J}$ consists of two parts, i.e. the Jacobian of $F(\mathbf{x})$ w.r.t. the poses $\mathbf{J}_P$, and the Jacobian of $F(\mathbf{x})$ w.r.t. the global occupancy map $\mathbf{J}_M$, i.e. $\mathbf{J} = [\mathbf{J}_P \quad \mathbf{J}_M]$. All non-zero elements of $\mathbf{J}_M$ are $\omega(i, \mathbf{m}_{wh})$ which only depend on poses (hit maps N^M, N^{L_i} are independent of occupancy values), and thus $\mathbf{J}_M$ is independent of $\mathbf{x}^M$. In addition, it can be deducted from (4.1) and (4.4) that $\mathbf{J}_P$ is also independent of $\mathbf{x}^M$.

Let us rewrite (4.6) as

$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^\top & \mathbf{V} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}^P \\ \boldsymbol{\Delta}^M \end{bmatrix} = \begin{bmatrix} \mathbf{b}^P \\ \mathbf{b}^M \end{bmatrix} \tag{4.7}$$

where $\mathbf{U} = \mathbf{J}_P^\top \mathbf{J}_P$, $\mathbf{V} = \mathbf{J}_M^\top \mathbf{J}_M$, $\mathbf{W} = \mathbf{J}_P^\top \mathbf{J}_M$, $\mathbf{b}^P = -\mathbf{J}_P^\top F(\mathbf{x})$, and $\mathbf{b}^M = -\mathbf{J}_M^\top F(\mathbf{x})$. According to the theory of the Schur complement [131], the solution of (4.7) can be calculated by

$$\left(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^\top\right) \boldsymbol{\Delta}^P = \mathbf{b}^P - \mathbf{W}\mathbf{V}^{-1}\mathbf{b}^M \tag{4.8}$$

and

$$\mathbf{V}\boldsymbol{\Delta}^M = \mathbf{b}^M - \mathbf{W}^\top \boldsymbol{\Delta}^P. \tag{4.9}$$

If we rewrite (4.8) as

$$\begin{aligned} &\left(\mathbf{J}_P^\top \mathbf{J}_P - \mathbf{J}_P^\top \mathbf{J}_M \left(\mathbf{J}_M^\top \mathbf{J}_M\right)^{-1} \mathbf{J}_M^\top \mathbf{J}_P\right) \boldsymbol{\Delta}^P \\ =& -\mathbf{J}_P^\top F(\mathbf{x}) + \mathbf{J}_P^\top \mathbf{J}_M \left(\mathbf{J}_M^\top \mathbf{J}_M\right)^{-1} \mathbf{J}_M^\top F(\mathbf{x}), \end{aligned} \tag{4.10}$$

it can be seen that the left-hand side of (4.10) is not related to $\mathbf{x}^M$ because both $\mathbf{J}_M$ and $\mathbf{J}_P$ are independent of $\mathbf{x}^M$.

Furthermore, we can rewrite (4.4) as

$$f(\mathbf{x}) = \|F(\mathbf{x})\|^2 = \|\mathbf{J}_M\mathbf{x}^M - H(\mathbf{x}^P))\|^2, \tag{4.11}$$

where $H(\mathbf{x}^P) = [\cdots, L_i(\mathbf{p}_i^{wh}), \cdots]$ is not related to $\mathbf{x}^M$. Using $F(\mathbf{x})$ in (4.11), the right-hand side of (4.10) becomes

$$\begin{aligned} &-\mathbf{J}_P^\top(\mathbf{J}_M\mathbf{x}^M - \mathbf{J}_M(\mathbf{J}_M^\top\mathbf{J}_M)^{-1}\mathbf{J}_M^\top\mathbf{J}_M\mathbf{x}^M) \\ &-\mathbf{J}_P^\top(\mathbf{J}_M(\mathbf{J}_M^\top\mathbf{J}_M)^{-1}\mathbf{J}_M^\top H(\mathbf{x}^P) - H(\mathbf{x}^P)). \end{aligned} \tag{4.12}$$

It can be deduced that the first item of (4.12) is equal to 0, and the second item is not related to $\mathbf{x}^M$ because $\mathbf{J}_M$, $\mathbf{J}_P$ and $H(\mathbf{x}^P)$ are all independent of $\mathbf{x}^M$. Thus, the right-hand side of (4.10) is not related to $\mathbf{x}^M$. Therefore, the increment of poses $\boldsymbol{\Delta}^P$ is independent of $\mathbf{x}^M$ and it can be calculated by

$$\begin{aligned} &\left(\mathbf{J}_P^\top\mathbf{J}_P - \mathbf{J}_P^\top\mathbf{J}_M(\mathbf{J}_M^\top\mathbf{J}_M)^{-1}\mathbf{J}_M^\top\mathbf{J}_P\right)\boldsymbol{\Delta}^P \\ &= \mathbf{J}_P^\top H(\mathbf{x}^P) - \mathbf{J}_P^\top\mathbf{J}_M(\mathbf{J}_M^\top\mathbf{J}_M)^{-1}\mathbf{J}_M^\top H(\mathbf{x}^P). \end{aligned} \tag{4.13}$$

□

For the point feature-based SLAM problem with known data association and under the assumption that the observation noises covariance matrices are isotropic, a similar property was proved in [132]. In this thesis, we prove that this independent property also holds in the proposed grid-based submap joining problem without the assumption of data association.

4.2.3 Equivalent Pose-only GN Iteration Algorithm

According to the Proposition proved in Section 4.2.2, we propose a pose-only GN algorithm equivalent to full GN method to solve the proposed NLLS problem. The equivalent pose-only GN algorithm is given in Algorithm 5. In Algorithm 5, we iteratively solve (4.13) using Jacobian $\mathbf{J}_P$ and $\mathbf{J}_M$ to obtain the increments of the poses $\boldsymbol{\Delta}^P$ and update poses,

Algorithm 5: Pose-only GN Algorithm

Input: Submaps $\mathbb{L}$, local hit maps $\mathbb{N}^L$, and initial poses $\mathbf{x}^P(0)$
Output: Optimized poses $\hat{\mathbf{x}}^P$ and optimized global occupancy map $\hat{\mathbf{x}}^M$

```
Initialize global occupancy map x^M(0) and global hit map N^M(0) using x^P(0), 𝕃 and 𝕹^L;
for k = 0 to τ_k and ‖Δ^P(k)‖² ≥ τ_Δ^P do
    Evaluate H(x^P) and Jacobians J_P, J_M at x^P(k);
    Solve linear system (4.13) to get Δ^P(k);
    Update poses: x^P(k+1) = x^P(k) + Δ^P(k);
    Recalculate global hit map N^M(k+1) using x^P(k+1) and 𝕹^L;
Set x̂^P = x^P(k+1);
Solve closed-form formula (4.15) to get optimized global occupancy map x̂^M;
```

and then recalculate the global hit map N^M based on the updated poses since $\mathbf{J}_M$ depends on the global hit map. After the optimal solution of poses $\hat{\mathbf{x}}^P$ is obtained, the proposed NLLS problem (4.4) becomes a linear least squares problem which minimize

$$g(\mathbf{x}^M) = (\mathbf{J}_M\mathbf{x}^M - H(\hat{\mathbf{x}}^P))^\top(\mathbf{J}_M\mathbf{x}^M - H(\hat{\mathbf{x}}^P)). \tag{4.14}$$

Therefore, the optimal solution of the global occupancy map $\hat{\mathbf{x}}^M$ can be calculated by the closed-form formula

$$\hat{\mathbf{x}}^M = \mathbf{V}^{-1}\mathbf{J}_M H(\hat{\mathbf{x}}^P). \tag{4.15}$$

Here, $\mathbf{V} = \mathbf{J}_M^\top\mathbf{J}_M$ is the map-related Hessian block, as defined in (4.7).

For Algorithm 5, the dimension of the sparse linear system to be solved at each iteration only depends on the number of poses, whereas in the full GN algorithm, it depends on the number of poses and the number of cells in the global map. For the grid-based submap joining problem in which the number of poses is much less than that of cells of the global map, the proposed pose-only GN algorithm can easily and quickly obtain the solution of $\boldsymbol{\Delta}^P$ for each iteration. After optimal poses $\hat{\mathbf{x}}^P$ are obtained, the optimized global map $\hat{\mathbf{x}}^M$ can be easily obtained by the closed-form formula (4.15).

It should be noted that, although our proposed pose-only GN does not need to update the map during iterations, the constraint information associated with the map is still implicitly taken into account when optimizing the poses. Thus, the proposed approach is quite different from pose-graph optimization.

4.3 Experimental Results

In this section, we demonstrate the accuracy and efficiency of the proposed algorithm in this Chapter, hereafter Occupancy-Joining. In the experiments, we compare results of Occupancy-Joining with Cartographer [8], SLAM Toolbox [130], and Occupancy-SLAM [15] [1], which are the current state-of-the-art algorithms and perform significantly better than other methods such as Hector-SLAM [14], Karto-SLAM [67], etc.

First, due to the lack of ground truth in the practical datasets, we evaluate Occupancy-Joining qualitatively and quantitatively using datasets generated from two simulated environments. Second, we validate and evaluate the performance of Occupancy-Joining through six publicly available practical datasets, including three datasets in large-scale environments. Parameters of all used datasets are listed in Table 4.1. All OGMs are generated by poses from different approaches and corresponding observations using the same evidence grid mapping technique. In addition, we use Occupancy-SLAM to build local submaps used in Occupancy-Joining for all the experiments, in which the map resolution is the default setting in [15] for simulation experiments and low-resolution for practical experiments.

TABLE 4.1: Parameters of Datasets.

Dataset	No. Scans	Duration (s)	Map Size (m)	Odometry
Simulation 1	3640	117	50×50	yes
Simulation 2	2680	83	50×50	yes
Car Park [133]	1642	164	50×40	yes
C5 [15]	3870	136	50×40	yes
Museum b0 1G [8]	5522	152	85×95	no
Museum b2 [8]	51833	1390	250×200	no
Museum b0 EG [8]	22650	615	225×150	no
C3	24402	610	150×125	no

[1] This work presented in this chapter is only a partial version of the proposed full method presented in Chapter 3, and exhibits lower efficiency, reduced convergence performance, and decreased robustness compared to the complete approach.

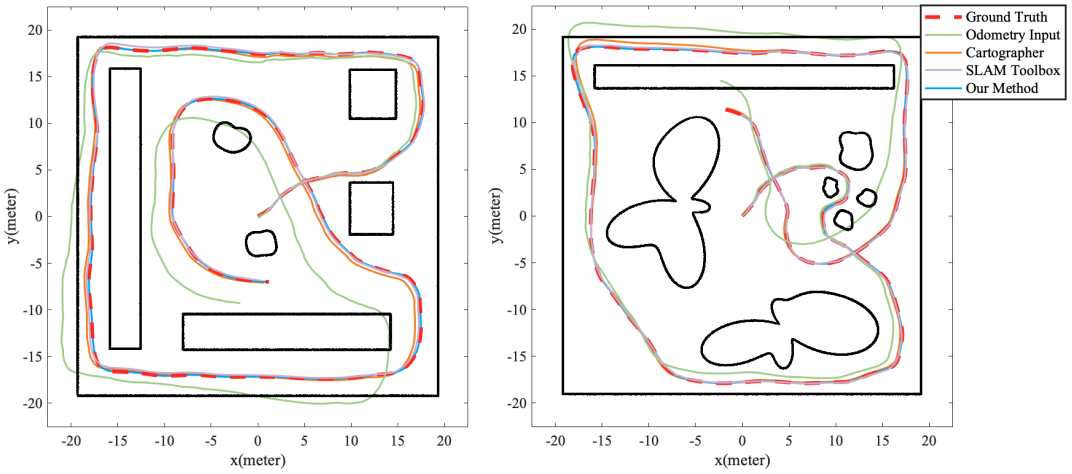
4.3.1 Simulation Experiments

We use varying levels of nonlinearity, non-convex obstacles, and long corridors to design two different simulation experiments. For all simulated datasets, each scan consists of 1081 laser beams with angles ranging from -135 degrees to 135 degrees which simulates a Hokuyo UTM-30LX laser scanner. To simulate real-world data acquisition, we add random Gaussian noises with zero-mean and standard deviation of 0.02 m to each beam of the scan data generated from the ground truth. Similarly, we add zero-mean Gaussian noises to the odometry inputs generated from the ground truth poses (standard deviation of 0.04 m for x-y and 0.003 rad for orientation). For each simulation environment, we generate 5 datasets with different sets of random noises.

The robot trajectory estimation results of Occupancy-Joining, SLAM Toolbox and Cartographer of one dataset in each simulation are compared with the ground truth and odometry in Figure 4.2. Since no difference can be visualized between trajectory of Occupancy-Joining and the trajectory from Occupancy-SLAM, the trajectory from Occupancy-SLAM is not drawn. It is obvious that trajectories of Occupancy-Joining are closer to the ground truth trajectories.

The quantitative comparison of errors in the estimated poses from Cartographer, SLAM Toolbox, Occupancy-SLAM and Occupancy-Joining is given in Table 4.2, in which we use MAE and RMSE to evaluate the translation errors (in meters) and rotation errors (in radians) for the 5 runs in each simulation. It should be noted that for SLAM Toolbox, since only the poses of submap frames are involved in the pose graph optimization, here we only use those poses of the submap frames to evaluate the pose errors. Compared to Cartographer and SLAM Toolbox, Occupancy-Joining performs better on all metrics, with errors about 3 to 12 times smaller. The significant reduction in error validates the effectiveness of Occupancy-Joining. Occupancy-SLAM performs best in terms of accuracy due to the use of the full least squares strategy, which is the best one can achieve but is costly.

Figure 4.3 shows the OGMs and point cloud maps generated by poses from ground truth, Cartographer, SLAM Toolbox, Occupancy-SLAM and Occupancy-Joining, where the first two rows are the results of a dataset in Simulation 1, and the last two rows are the results



(a) Simulation 1 (b) Simulation 2

FIGURE 4.2: **Simulation environments and robot trajectory results.** (a) and (b) show the simulation environments (the black lines indicate the obstacles in the scene) and the trajectories of ground truth, odometry inputs, Cartographer [8], SLAM Toolbox [130], and Occupancy-Joining for one dataset in each of the two simulation experiments. Results from Occupancy-SLAM [15] are visually identical to Occupancy-Joining and not drawn in the figure.

of a dataset in Simulation 2. The areas highlighted by red dots show that the results of Occupancy-Joining are better than the results of Cartographer and SLAM Toolbox. It is clear that the object boundaries of both OGMs and point cloud maps obtained by Occupancy-Joining are much clearer than Cartographer and SLAM Toolbox. Also, it can be seen that the maps of Occupancy-Joining are very close to Occupancy-SLAM. This suggests that Occupancy-Joining, as an approximation scheme of full optimization based SLAM, can obtain similar results as the full optimization based approach Occupancy-SLAM.

For the quantitative comparison of occupancy maps, we use the results from all the ten datasets in Simulation 1 and Simulation 2 to compare the accuracy of the OGMs. We use AUC and precision to evaluate Occupancy-Joining, Cartographer, SLAM Toolbox and Occupancy-SLAM, where ground truth labels are given by OGMs generated using ground truth poses. When evaluating using AUC, we remove all the unknown cells because AUC is a metric for binary classification as in [124]. The results are given in Table 4.3. Compared with Cartographer and SLAM Toolbox, it can be seen that Occupancy-Joining achieves better performance in both metrics, which proves Occupancy-Joining can obtain more

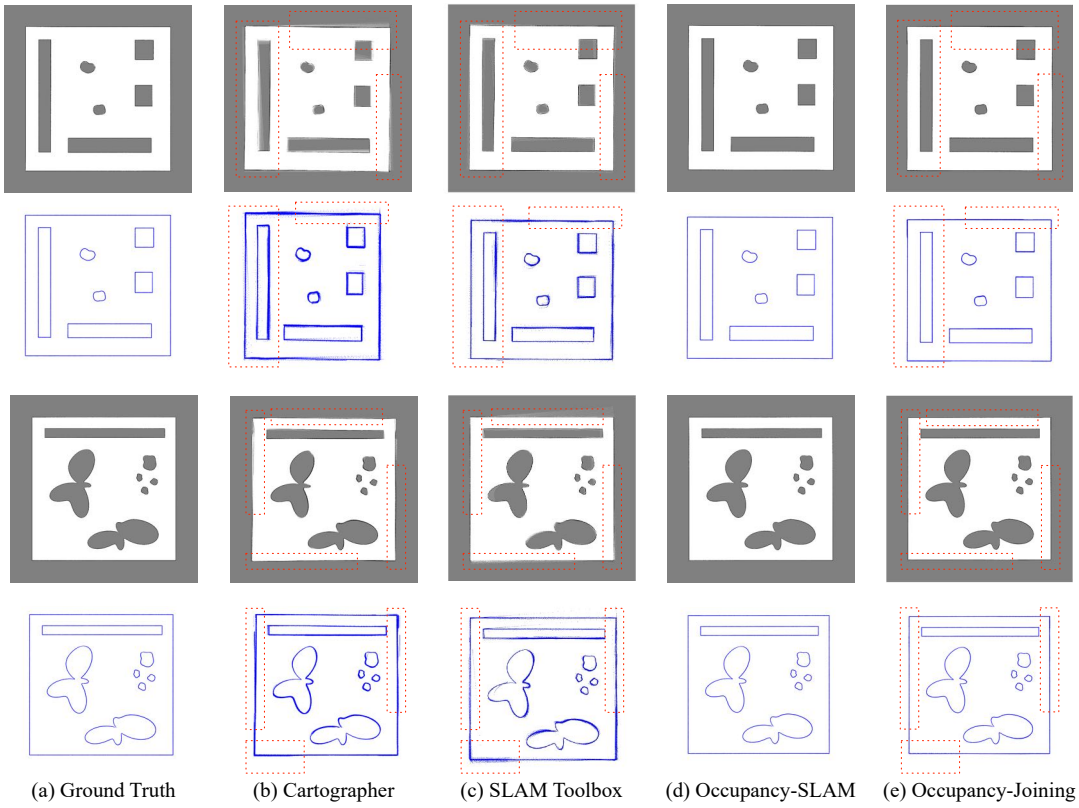


FIGURE 4.3: **OGMs and point cloud maps comparisons in simulation.** The OGMs and point cloud maps generated by poses from ground truth, Cartographer, SLAM Toolbox, Occupancy-SLAM and Occupancy-Joining in one dataset for Simulation 1 (first two rows) and Simulation 2 (last two rows).

accurate maps.

4.3.2 Comparisons Using Practical Datasets

First, we use three practical datasets used in [15], namely *Deutsches Museum b0 1G* [8], *Car Park* [133] and *C5* [15], to compare Occupancy-Joining with Cartographer and Occupancy-SLAM. The results are shown in Figure 4.4 and Figure 4.5. It is clear that results of Occupancy-Joining are very close to those of Occupancy-SLAM. Compared with Cartographer's results, both the OGMs and point cloud maps of Occupancy-Joining have significantly clearer object boundaries. These results show the accuracy of Occupancy-Joining using practical datasets. Since SLAM Toolbox does not support the *MultiEchoLaserScan* sensor message format of Cartographer's companion *Deutsches Museum* dataset [8] and performs poorly on *Car Park* and *C5* datasets because no loop closures are

TABLE 4.2: Comparison of Robot Pose Errors in Simulations.

	Odom	Carto	Toolbox	Occupancy-SLAM*	Occupancy-Joining
Simulation 1					
MAE (Trans/m)	0.7827	0.2534	0.1465	0.0238	0.0453
MAE (Rot/rad)	0.0491	0.0139	0.0134	0.0008	0.0012
RMSE (Trans/m)	0.9840	0.2992	0.1788	0.0263	0.0609
RMSE (Rot/rad)	0.0551	0.0156	0.0152	0.0009	0.0015
Simulation 2					
MAE (Trans/m)	0.7535	0.1426	0.0493	0.0070	0.0151
MAE (Rot/rad)	0.0518	0.0068	0.0085	0.0006	0.0011
RMSE (Trans/m)	0.9687	0.1878	0.0626	0.0106	0.0205
RMSE (Rot/rad)	0.0593	0.0091	0.0127	0.0009	0.0018

Red and the blue indicate the best and second best results respectively.
* Occupancy-SLAM uses the full least squares strategy including all the robot poses, which can theoretically achieve the highest accuracy. However, it is very costly and cannot handle large-scale environments. For Simulation 1 and Simulation 2, Occupancy-SLAM takes about 20 times longer than Occupancy-Joining. The time consumptions on some practical datasets are given in Table. 4.4.

TABLE 4.3: Accuracy of the Occupancy Grid Maps.

		AUC	Precision
Simulation 1	Cartographer	0.9088	0.9565
	SLAM Toolbox	0.9550	0.9749
	Occupancy-SLAM	0.9997	0.9973
	Occupancy-Joining	0.9751	0.9867
Simulation 2	Cartographer	0.9270	0.9660
	SLAM Toolbox	0.9726	0.9871
	Occupancy-SLAM	0.9966	0.9991
	Occupancy-Joining	0.9949	0.9979

detected to perform pose graph optimization, we do not present the results of the SLAM Toolbox for this and subsequent experiments.

Occupancy-Joining can handle large-scale environments well, which is impossible with Occupancy-SLAM due to its increasing computational complexity and tendency to fall into local minima as the trajectory and map size grow. To validate the effectiveness of Occupancy-Joining in large-scale environments, three large-scale environment datasets are utilized, namely *Deutsches Museum b0 EG*, *Deutsches Museum b2*, and *C3*, and their map sizes range from 150 m $\times$ 125 m to 250 m $\times$ 200 m. Here, we only compared Occupancy-Joining with Cartographer, the results of OGMs are shown in Figure 4.6, and it can be

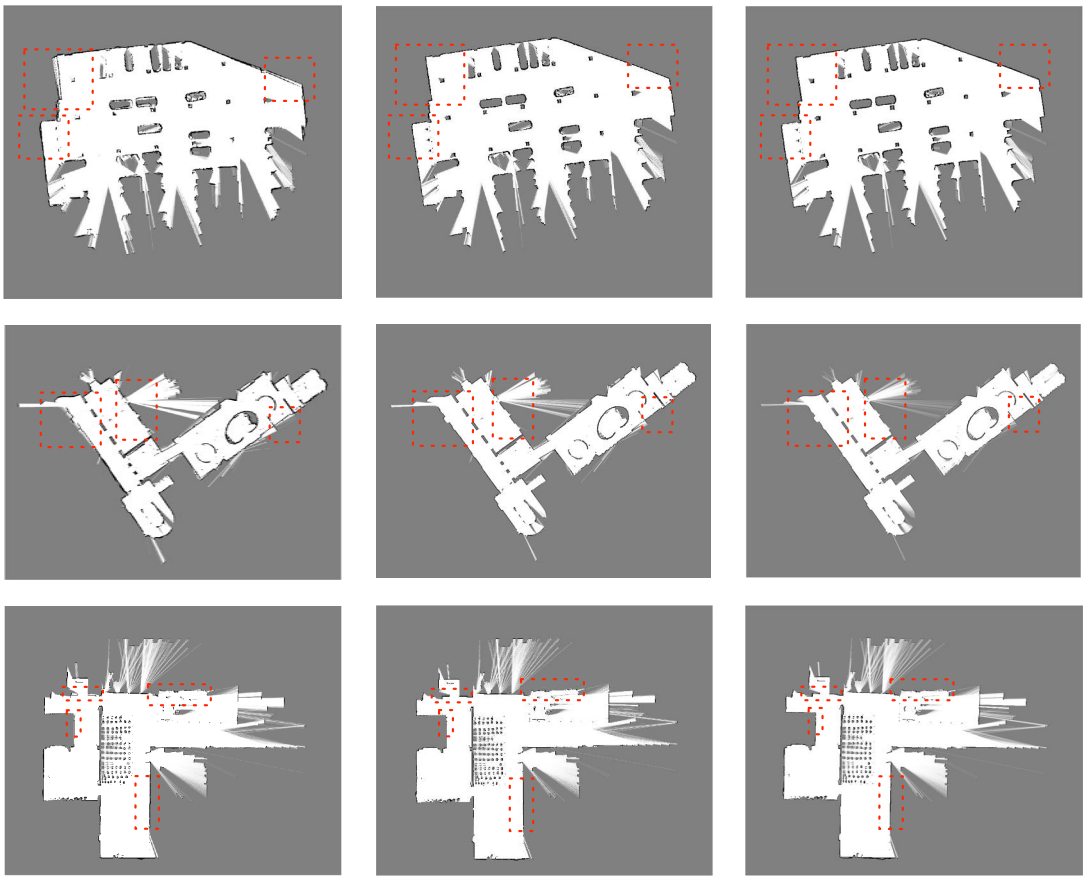


FIGURE 4.4: **OGMs from Cartographer, Occupancy-SLAM and Occupancy-Joining.** The first to third rows are *Car Park*, *Deutsches Museum b0 1G* and *C5* dataset, respectively.

observed that results of Occupancy-Joining are better than Cartographer, especially for Figure 4.6(c) and Figure 4.6(f), where the right wall from results of Occupancy-Joining are clearly closer to the straight line whereas that of the Cartographer's results are somewhat curved.

4.3.3 Time Consumption

We use all the six practical datasets to evaluate the time consumption of Occupancy-Joining, Cartographer and Occupancy-SLAM, the results are presented in Table 4.4 (all

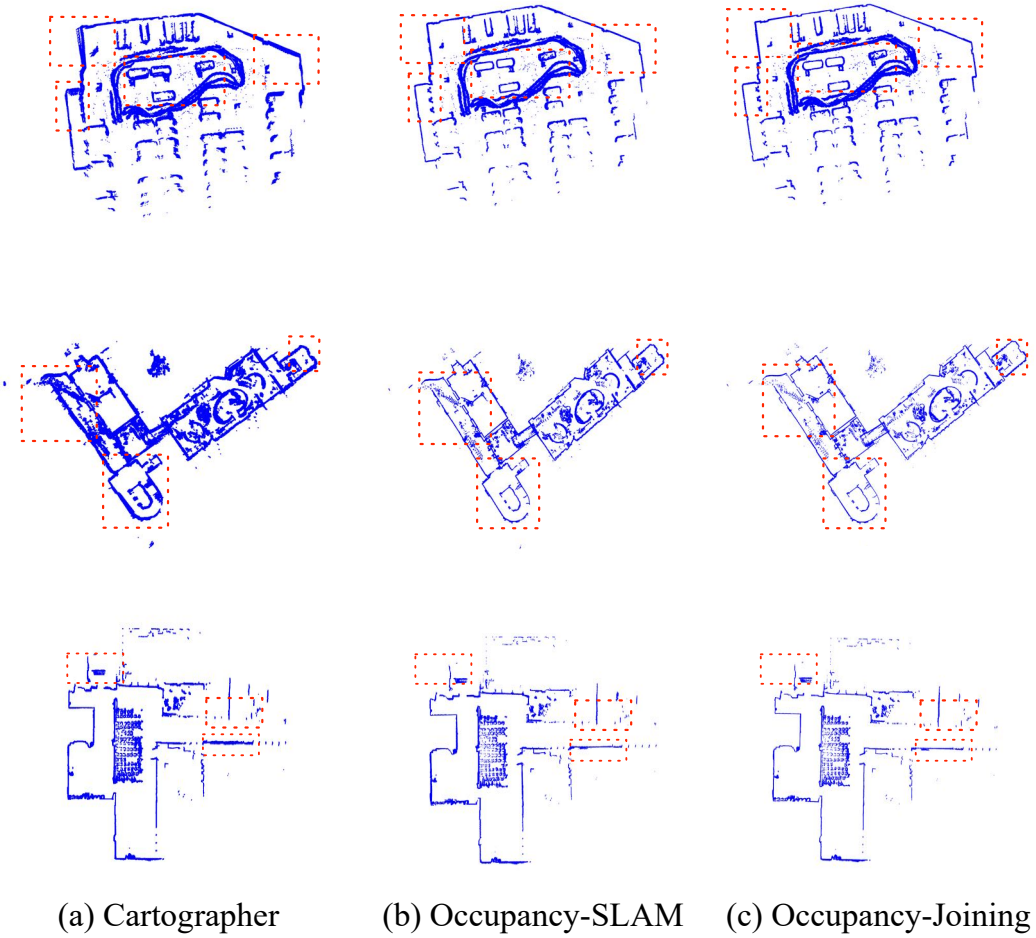
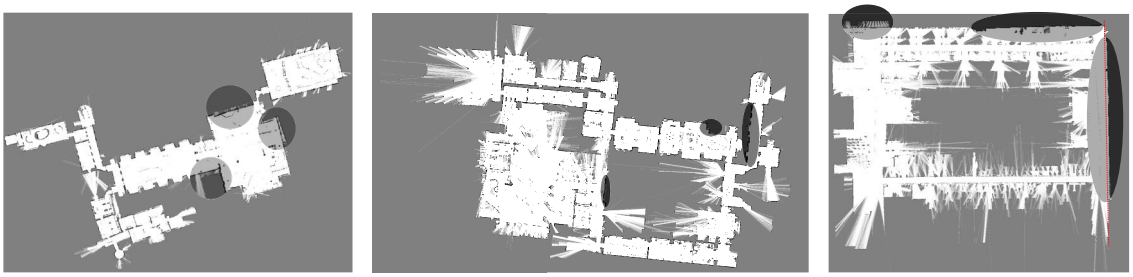
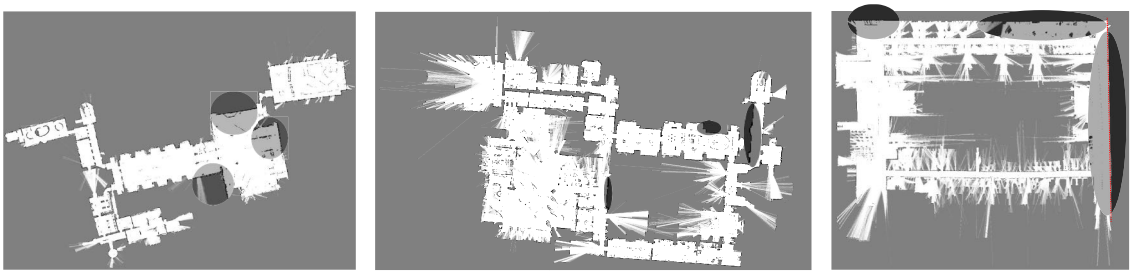


FIGURE 4.5: **Point cloud maps from Cartographer, Occupancy-SLAM and Occupancy-Joining.** The point cloud maps are generated by projecting the scan endpoints using poses.

elapsed times are evaluated using an Intel Core i7-1370P processor). The huge time consumption of Occupancy-SLAM is due to the high computational complexity of batch optimization using all the scans. Thus, it cannot handle the three large-scale datasets. In contrast, Occupancy-Joining is very efficient, taking less than 1 second for the three normal-scale datasets and around 2 to 6 seconds for the three large-scale datasets. Since Cartographer and Occupancy-SLAM can solve the complete OGM based SLAM problems from scans, we also list the elapsed time of submap building. The total time of submap building and joining is still faster than that of Cartographer on most datasets. It should be mentioned that this total time consumption can be further reduced by using other efficient submap building methods, and the submaps can also be built out-of-core as in [134].



(a) b2 Cartographer (b) b0 EG Cartographer (c) C3 Cartographer



(d) b2 Occupancy-Joining (e) b0 EG Occupancy-Joining (f) C3 Occupancy-Joining

FIGURE 4.6: **Comparison of the results of Occupancy-Joining and Cartographer in three large-scale environments.** The highlighted areas show Occupancy-Joining outperforms Cartographer. The red dotted lines in (c) and (f) are reference lines, and it can be seen that the right wall of Cartographer's result is more curved while result of Occupancy-Joining is closer to the straight line. Occupancy-SLAM fails with such large-scale environments.

TABLE 4.4: Time Consumption (*in seconds*) of Algorithms

Dataset	Carto	Occupancy-SLAM	Occupancy-Joining		
			Submap	Joining	Total
Car Park	168	17138	133	0.8	133.8
Museum b0 1G	152	26901	113	0.6	113.6
C5	146	27452	157	0.6	157.6
Museum b2	1424	-	1248	2.4	1250.4
Museum b0 EG	615	-	539	1.7	540.7
C3	610	-	737	5.2	742.2

The high efficiency of Occupancy-Joining is mainly due to the independent property of the proposed formulation. Because of this, the pose-only GN method can be used to solve the NLLS problem efficiently. We compare our proposed pose-only GN with the full GN where both poses and global map are solved at the same time, and the results are depicted in Table 4.5. It is clear that the calculation speed of our proposed pose-only algorithm is much faster than the full GN algorithm.

TABLE 4.5: Time Consumption (*in seconds*) of Each Iteration*

Dataset	Pose-only GN	Full GN	Acceleration Rate
Car Park	0.062	0.174	281%
Museum b0 1G	0.044	0.098	223%
C5	0.036	0.058	161%
Museum b0 EG	0.157	0.419	267%
Museum b2	0.226	0.378	167%
C3	0.096	0.174	181%

*Since the proposed pose-only GN algorithm is equivalent to the full GN algorithm and thus has the same number of iterations, we only compare the time consumption per iteration.

4.3.4 Computational Complexity Analysis

In this section, we first analyze the computation complexity of full GN for our proposed grid-based submap joining problem. Then we analyze the computational complexity of our proposed pose-only GN algorithm for solving this problem.

When the full GN is applied to solve the proposed grid-based submap joining problem, it solves the linear system defined in (4.6). Thus, the computational cost of the full GN for solving submap joining problem is primarily arises from three key steps: computing the Jacobian matrix $\mathbf{J}$, constructing, and solving the sparse linear system in (4.6). The number of observations depends on the total number of cell vertices of the global occupancy map observed in each submap, denoted $\mathfrak{d}_{obs}^{G}$. Considering that some cell vertices will be observed repeatedly under different submaps, this number slightly exceeds the number of non-unknown cell vertices in the global map. We simplify the notation by defining the global map size as $\mathfrak{d}_{M}^{G} = (c_{w}^{G}+1)(c_{h}^{G}+1)$, and the number of pose variables in the state vector as $\mathfrak{d}_{P}^{G} = 3n_{L}$. Therefore, the number of non-zero elements of Jacobian matrix is $\mathfrak{d}_{J}^{G} = 4\mathfrak{d}_{obs}^{G}$, and the state vector size is $\mathfrak{d}_{col}^{G} = \mathfrak{d}_{P}^{G} + \mathfrak{d}_{M}^{G}$. Consequently, the per-iteration computation complexities of the Jacobian calculation, constructing (4.6) and solving (4.6) are $\mathfrak{d}_{J}^{G}$, $\mathfrak{d}_{J}^{G}\mathfrak{d}_{col}^{G}$, and ${\mathfrak{d}_{col}^{G}}^{3}$, respectively. The computation complexity per iteration is $\mathcal{O}(\mathfrak{d}_{J}^{G} + \mathfrak{d}_{J}^{G}\mathfrak{d}_{col}^{G} + {\mathfrak{d}_{col}^{G}}^{3}) = \mathcal{O}({\mathfrak{d}_{col}^{G}}^{3})$. Furthermore, as the sub-matrix of Hessian w.r.t. the global occupancy map is diagonal, utilizing the Schur complement [131] can significantly enhance the efficiency of solving this joint optimization problem.

When applying our proposed pose-only GN method to solve the submap joining problem, the linear system (4.13) is solved at each iteration. For simplicity, we represent (4.13) as $\mathbf{A}\boldsymbol{\Delta}^P = \mathbf{b}$. The computation cost of the pose-only GN primarily involves computing the matrix $\mathbf{A}$, computing the vector $\mathbf{b}$, and solving the sparse linear system (4.13). Given that $\mathbf{J}_M^\top \mathbf{J}_M$ is a diagonal sparse matrix, calculating its inverse $\left(\mathbf{J}_M^\top \mathbf{J}_M\right)^{-1}$ is computationally inexpensive. Thus, the primary computational cost in computing $\mathbf{A}$ is $\mathcal{O}(\mathfrak{d}_{obs}^G \mathfrak{d}_P^{G2} + \mathfrak{d}_{obs}^G \mathfrak{d}_P^G \mathfrak{d}_M^G + \mathfrak{d}_P^{G2} \mathfrak{d}_M^G)$. The complexity for constructing vector $\mathbf{b}$ is $\mathcal{O}(\mathfrak{d}_{obs}^G \mathfrak{d}_P^G + \mathfrak{d}_{obs}^G \mathfrak{d}_M^G + \mathfrak{d}_P^G \mathfrak{d}_M^G)$, and solving the sparse linear system (4.13) is $\mathcal{O}(\mathfrak{d}_P^{G3})$. Considering typical scenarios in submap joining problems, the number of poses is substantially smaller than the map size and the number of observations. Hence, the dominant complexity term becomes $\mathcal{O}(\mathfrak{d}_{obs}^G \mathfrak{d}_M^G \mathfrak{d}_P^G)$. Clearly, the computational complexity of our proposed pose-only GN approach is significantly lower than that of the full GN approach.

4.4 Experimental Results in 3D Case

4.4.1 Extension of the Algorithms to 3D Case

The proposed approach for jointly optimizing robot poses and the occupancy map extends naturally to 3D, where the information, robot poses, and occupancy maps are all represented in 3D. Most problem formulations and algorithms can be adapted with minor adjustments.

The submap joining problem in 3D remains largely similar to the 2D case, except that the projection relation extends from 2D-2D to 3D-3D, enabling the solution of 6 DoF poses and the 3D global occupancy map in the NLLS problem (4.6).

4.4.2 3D Experimental Results

4.4.2.1 Evaluation Metrics and State-of-the-art Methods

We evaluate the performance of Occupancy-Joining in 3D using ATE for poses, aligning and comparing results with ground truth via EVO [126], as used in [78, 116]. In all the

experiments, we use the odometry information provided by the dataset as initialization if it is available. Otherwise, we use FAST-LIO2 [127] to obtain the odometry information. To evaluate Occupancy-Joining, we compare Occupancy-Joining against state-of-the-art methods: HBA [116] and Voxgraph [10]. HBA proposes a hierarchical bundle adjustment to optimize the consistency of the planar surfaces of point clouds and robot poses. Voxgraph builds SDF-based submaps from point clouds, uses SDF-to-SDF registration for relative submap measurements, and incrementally optimizes submap frames.

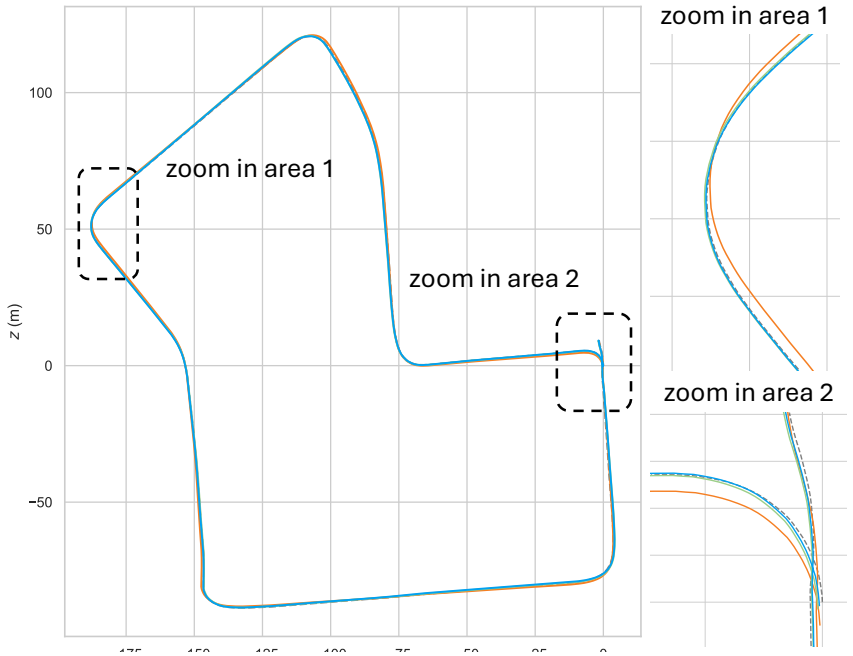
4.4.2.2 Datasets

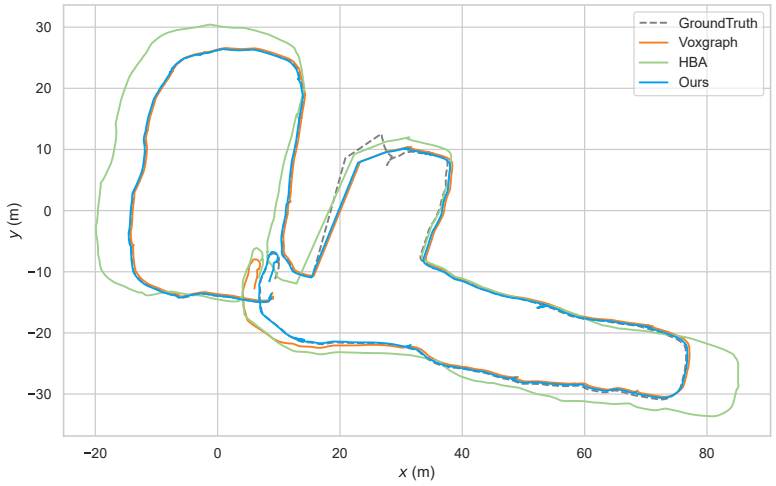
We perform comparisons using two real-world datasets. (1) KITTI Dataset [135]: Sequence 07, a demo dataset for HBA, collected with a Velodyne HDL-64E LiDAR scanner mounted on a car. Ground truth poses are provided by RTK-GPS/INS. (2) Arche Dataset [10]: Details are provided in Section 3.5.2.2.

4.4.2.3 Experiments

We evaluate Occupancy-Joining in large-scale environments with long trajectories using the KITTI and Arche datasets, comparing it with HBA and Voxgraph. For this experiment, we first apply the Occupancy-SLAM method [16], as detailed in Chapter 3, to optimize robot poses within each local submap. Based on the optimized poses, submaps are constructed, and Occupancy-Joining is then applied to jointly optimize the submap frame poses and the global occupancy map.

Table 4.6 summarizes the MAE and RMSE of ATE, showing Occupancy-Joining achieves the best results on both datasets. Figure 4.7 illustrates that Occupancy-Joining can achieve the best global robot trajectories. Notably, Occupancy-Joining significantly outperforms Voxgraph on the KITTI dataset and HBA on the Arche dataset. The relatively poor performance of Voxgraph on the KITTI dataset is due to its reliance on relative measurements from SDF-to-SDF registration, which requires sufficient overlapping submaps—a challenge in autonomous driving scenarios. In contrast, Occupancy-Joining jointly optimizes submap poses and the global occupancy map, avoiding this limitation. HBA underperforms on the





(b) Arche

FIGURE 4.7: **Robot trajectory results of datasets in large-scale environments.** (a) and (b) show the trajectories of ground truth, Voxgraph [10], HBA [116], and Occupancy-Joining for KITTI dataset and Arche dataset.

Arche dataset due to its reliance on planar features for optimization, which is challenging in unstructured environments and during MAV motion. Odometry and point clouds from such scenarios make detecting sufficient planar features difficult, and the planarity assumption often fails in non-urban environments like disaster areas.

TABLE 4.6: Absolute Trajectory Error (MAE/RMSE, Meters) in Large-scale Environments for Different 3D Methods

Method	KITTI	Arche
HBA [116]	0.342/0.364	4.123/4.789
Voxgraph [10]	0.926/1.002	0.700/0.833
Occupancy-Joining	**0.315/0.339**	**0.275/0.378**

4.5 Chapter Summary

This chapter formulates the grid-based submap joining problem as a NLLS problem to simultaneously optimize the global occupancy map and local submap frames, which is different from other submap joining approaches that only optimize poses by constructing a pose graph. More importantly, we show that for the proposed formulation, the existence of a special independent property helps solve this problem very efficiently by a pose-only GN algorithm. The proposed submap joining approach is evaluated using datasets generated from two simulated environments and six practical datasets in the 2D case, demonstrating that it outperforms state-of-the-art algorithms in terms of speed and accuracy. Furthermore, we extend the method to 3D scenarios and validate it using two real-world datasets, showing improved accuracy over existing state-of-the-art approaches.

The proposed joint optimization of poses and occupancy maps has been shown to outperform previous two-step decoupled approaches in OGM-based SLAM methods for both normal-scale and large-scale scenarios. It is worth further investigating whether this joint optimization framework is also suitable for other non-feature-based SLAM problems.

Chapter 5

Joint Optimization of Robot Poses and Signed Distance Function Map

In Chapters 3 and 4, we demonstrated that jointly optimizing an occupancy map and robot poses can yield more accurate results than state-of-the-art methods. Beyond OGMs, SDF maps are another popular non-feature-based representation widely used in SLAM. An SDF models the environment as a continuous signed distance field, where the zero-level set precisely defines the surface. This property not only supports high-quality 3D reconstruction but also provides a rich geometric structure that benefits downstream robotic tasks. Moreover, the continuous nature of the distance field offers smoother gradients than OGMs, which can be leveraged to guide optimization well. Given these advantages, it is worth investigating whether joint optimization of poses and an SDF map is similarly feasible and effective.

To this end, we propose a novel framework that jointly optimizes robot poses and a TSDF map. The method leverages the intrinsic properties of the SDF field to formulate surface, space, and consistency constraints across the TSDF volume. Unlike existing SDF-based approaches that rely on point-to-SDF or SDF-to-SDF registration, our formulation directly incorporates geometric and field-level constraints from the TSDF, enabling richer use of spatial information. In contrast to plane feature–based bundle adjustment methods that require explicit feature extraction and manual data association, our approach embeds

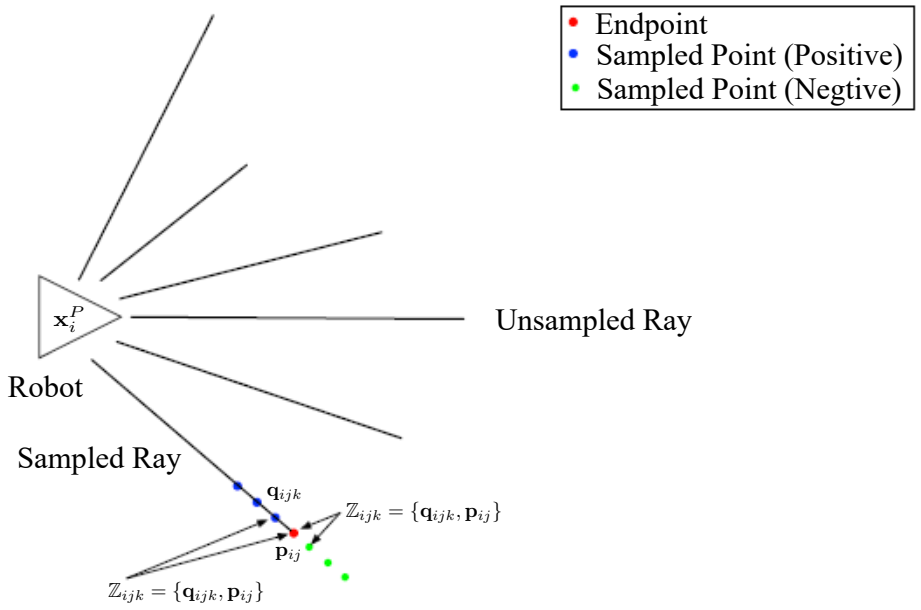


FIGURE 5.1: **Equidistant sampling strategy for generating observations from a LiDAR scan.** The distance between two adjacent sampled points is defined as the sampling step, while the total distance between the first and last sampled points corresponds to the truncation distance. All sampled point pairs constitute the observations.

consistency relationships implicitly within the TSDF representation. These relationships are dynamically updated during optimization, providing adaptive data association and improved robustness under noisy initializations.

Experimental results on real-world datasets show that the proposed method achieves superior robustness and accuracy compared to existing techniques, validating the effectiveness of jointly optimizing robot poses and TSDF maps.

5.1 Problem Formulation

5.1.1 Observations Information

We first define the observation information of the SDF-BA problem. Let the sequence of $n+1$ point clouds be denoted as $\mathbb{P} = \{\mathbb{P}_0, \ldots, \mathbb{P}_n\}$, where each point cloud $\mathbb{P}_i = \{\mathbf{p}_{ij}\}$ is expressed in its corresponding local coordinate frame. The pose of the i-th local frame is parameterized by $\mathbf{x}_i^P = [\mathbf{t}_i^\top, \xi_i^\top]^\top \in SE(3)$, where $\mathbf{t}_i$ is the 3D translation and ξ_i is a

rotation vector. The corresponding rotation matrix is given by $\mathbf{R}_i = \exp(\xi_i^\wedge)$, in which $\xi_i^\wedge$ denotes skew-symmetric matrix associated with the Lie algebra element ξ_i.

Each point $\mathbf{p}_{ij} \in \mathbb{P}_i$ in $\mathbb{P}_i$ is treated as the endpoint of a ray originating from the sensor. Along each such ray, we perform equidistant sampling at a step size r within a truncated distance τ_d from the hit point $\mathbf{p}_{ij}$, as illustrated in Figure 5.1. This results in $K = 2\tau_d/r$ sampled points denoted as $\{\mathbf{q}_{ijk}\}$. Each sampled point $\mathbf{q}_{ijk}$ is paired with its corresponding ray endpoint $\mathbf{p}_{ij}$ to form an observation pair $\mathbb{Z}_{ijk} = \{\mathbf{q}_{ijk}, \mathbf{p}_{ij}\}$. The complete observation set is thus organized as $\mathbb{Z} = \{\mathbb{Z}_{ijk}\}_{0 \le i \le n, 1 \le j \le |\mathbb{P}_i|, 1 \le k \le K}$, where i indexes LiDAR frames, j indexes rays (hit points) within a frame, and k indexes sampled points along each ray. Here, $|\mathbb{P}_i|$ denotes the number of points in the i-th point cloud. This set $\mathbb{Z}$ constitutes the full set of observations used in the SDF-BA optimization problem.

5.1.2 TSDF Map Representation

In the proposed method, we represent the environment using a TSDF, which models space as a field of signed distance values within a truncated region. Ideally, in a TSDF representation, a given point in space has a positive signed distance if it lies outside an object and a negative distance if it lies inside the object. The absolute value of the signed distance corresponds to the Euclidean distance to the closest surface.

In practice, the TSDF is discretized into voxels at a fixed resolution r, and the map is constructed by integrating a sequence of point clouds $\mathbb{P}$ along with their associated poses $\{\mathbf{x}_i^P\}$ using an approach similar to KinectFusion [48]. In the formulation, we denote $D(\cdot)$ as the TSDF value query function. It returns the signed distance value to its nearest surface directly when queried at a voxel index, and provides the value at an arbitrary 3D position by performing trilinear interpolation using the signed distance values of the eight surrounding voxel nodes.

5.1.3 State Vector

The proposed method aims to jointly optimize both the robot poses and the TSDF map. Therefore, the state vector $\mathbf{x}$ consists of the robot poses and all the voxel values within

the TSDF map, and is defined as

$$\mathbf{x} = \left[(\mathbf{x}^P),^\top, (\mathbf{x}^D)^\top\right]^\top, \tag{5.1}$$

where

$$\mathbf{x}^P = \left[(\mathbf{x}_0^P)^\top, \cdots, (\mathbf{x}_n^P)^\top\right]^\top \tag{5.2}$$

and

$$\mathbf{x}^D = [\cdots, D(\mathbf{m}_c), \cdots]^\top,\ c \in \Omega_{tsdf}. \tag{5.3}$$

In (5.3), $\mathbf{m}_c$ denotes the 3D coordinate of the c-th voxel, and Ω_{tsdf} represents the set of voxel indices that lie within the truncated signed distance region of the TSDF map.

5.1.4 Geometric Properties of Signed Distance Field

Before introducing our NLLS formulation, we outline several fundamental geometric properties of the signed distance field. These properties form the theoretical foundation of the constraints used in our optimization.

Property 1: Gradient Property For any point $\mathbf{p}$ in a signed distance field $D(\cdot)$, the gradient has unit magnitude:

$$\|\nabla D(\mathbf{p})\| = 1. \tag{5.4}$$

Since $\nabla D(\mathbf{p})$ points in the direction of the steepest increase of the signed distance value, its negative direction points toward the nearest surface. Thus, $-\nabla D(\mathbf{p})$ gives the direction from $\mathbf{p}$ to its closest surface point.

This unit-norm gradient property follows directly from the definition of a signed Euclidean distance.

Property 2: Surface Normal Property For any point $\mathbf{s}$ on the zero-level set of the SDF, i.e., $D(\mathbf{s}) = 0$, the outward-pointing surface normal equals the gradient of the SDF at that point:

$$\mathbf{n}_s = \nabla D(\mathbf{s}). \tag{5.5}$$

Because the zero-level set is an isosurface of the signed distance function, its gradient is orthogonal to the surface and naturally corresponds to the outward surface normal. Throughout the remainder of this paper, the term surface normal refers specifically to this outward-pointing normal.

Properties 1 and 2 lead to several immediate geometric consequences that are central to our constraint formulation. These are stated as the following lemmas.

Lemma 1: Closest-Surface Association. For any point $\mathbf{p}$ in a signed distance field $D(\cdot)$, there exists a unique closest surface point $\mathbf{s}$ on the zero-level set. This surface point can be computed directly from the SDF value and gradient at $\mathbf{p}$:

$$\mathbf{s} = \mathbf{p} - D(\mathbf{p})\,\nabla D(\mathbf{p}). \tag{5.6}$$

Lemma 2: Gradient Sharing. Let $\mathbf{s}$ be a surface point with outward unit normal $\mathbf{n}_s$. For any point $\mathbf{p}$ whose closest surface point is $\mathbf{s}$, the gradient at $\mathbf{p}$ in signed distance field is identical to its gradient at $\mathbf{s}$:

$$\nabla D(\mathbf{p}) = \nabla D(\mathbf{s}) = \mathbf{n}_s. \tag{5.7}$$

5.1.5 NLLS Formulation

Based on the state vector defined in (5.1) and the given lemmas in (5.6) and (5.7), we now formulate the NLLS problem to jointly optimize the robot poses and the TSDF map. The objective function of the NLLS problem is defined as

$$f(\mathbf{x}) = w_0 f^0(\mathbf{x}) + w_S f^S(\mathbf{x}) + w_{CP} f^C(\mathbf{x}) + w_N f^N(\mathbf{x}) + w_O f^O(\mathbf{x}). \tag{5.8}$$

The objective function consists of the surface constraint term $f^0(\mathbf{x})$, the space constraint term $f^S(\mathbf{x})$, the ray-surface-space consistency constraint term $f^C(\mathbf{x})$, the normalization constraint term $f^N(\mathbf{x})$, and the odometry term $f^O(\mathbf{x})$. w_0 , w_S, w_C, w_N, and w_O are their corresponding weights. We next explain the five terms individually.

1) Surface Constraint Term: The most straightforward constraint in the TSDF field is that the projected hit point

$$\mathbf{p}'_{ij} = \mathbf{R}_i \mathbf{p}_{ij} + \mathbf{t}_i. \tag{5.9}$$

should lie on the surface. In other words, its signed distance value $D(\mathbf{p}'_{ij})$ should be close to zero. Therefore, the surface constraint term in the objective function (5.8) is defined as

$$f^0(\mathbf{x}) = \sum_{i=0}^{N} \sum_{j} \|D(\mathbf{p}'_{ij})\|^2. \tag{5.10}$$

This term primarily constrains the robot poses and enforces the accuracy of the zero-level set of the TSDF voxels.

2) Space Constraint Term: To incorporate geometric consistency throughout the entire TSDF volume rather than only on the zero-level set, we introduce a space constraint term that extends the surface constraint into the surrounding region.

From **Lemma 1**, any spatial point in the TSDF field has a uniquely associated closest surface point that can be computed directly using its SDF value and gradient. Thus, for any voxel position $\mathbf{m}_c$ in the TSDF, its closest surface point is

$$\mathbf{s}_c = \mathbf{m}_c - D(\mathbf{m}_c) \nabla D(\mathbf{m}_c). \tag{5.11}$$

Since valid surface points must satisfy $D(\mathbf{s}_c) = 0$, we enforce this condition for all voxels via the following space constraint term:

$$\begin{aligned} f^S(\mathbf{x}) &= \sum_{c \in \Omega_{tsdf}} \|D(\mathbf{s}_c)\|^2 \\ &= \sum_{c \in \Omega_{tsdf}} \|D(\mathbf{m}_c - D(\mathbf{m}_c) \nabla D(\mathbf{m}_c))\|^2 . \end{aligned} \tag{5.12}$$

This term enforces global consistency in the TSDF field, ensuring that inferred surface points across the volume remain geometrically coherent with the underlying signed distance structure.

3) Ray-Surface-Space Consistency Constraint Term: The surface and space constraint terms rely primarily on observation endpoints and do not fully exploit the geometric relationships encoded by the sampled points along each LiDAR ray (or simulated ray from depth images). To make fuller use of the observation structure and to better constrain both robot poses and the TSDF field, we introduce the ray–surface–space consistency constraint. This term leverages the geometric linkage between each sampled point, its corresponding ray endpoint, and the surface geometry implied by the TSDF.

Each point $\mathbf{q}_{ijk}$ is paired with its corresponding ray endpoint $\mathbf{p}_{ij}$, and this pairing relationship is preserved under rigid body transformations:

$$\mathbf{q}'_{ijk} = \mathbf{R}_i \mathbf{q}_{ijk} + \mathbf{t}_i. \tag{5.13}$$

From **Lemma 1**, the closest surface point corresponding to $\mathbf{q}'_{ijk}$ in the TSDF field is

$$\mathbf{s}_{ijk} = \mathbf{q}'_{ijk} - D(\mathbf{q}'_{ijk}) \nabla D(\mathbf{q}'_{ijk}). \tag{5.14}$$

When $\mathbf{p}'_{ij}$ and $\mathbf{s}_{ijk}$ fall within locally consistent surface patches in the TSDF field, they share the same surface normal according to **Lemma 2**:

$$\mathbf{n}_s = \mathbf{n}_{p'_{ij}} = \mathbf{n}_{sijk} = \nabla D(\mathbf{s}_{ijk}). \tag{5.15}$$

Under this condition, their geometric relationship induces the ray–surface–space consistency constraint:

$$f^C(\mathbf{x}) = \sum_{i=0}^{n} \sum_{j} \sum_{k \in \Omega_C^i} \left\| \mathbf{n}_s{}^\top \left(\mathbf{s}_{ijk} - \mathbf{p}'_{ij} \right) \right\|^2, \tag{5.16}$$

where Ω_C^i denotes the set of indices for which $\mathbf{p}'_{ij}$ and $\mathbf{s}_{ijk}$ pass the consistency checks detailed in the next section. An illustration is given in Figure 5.2.

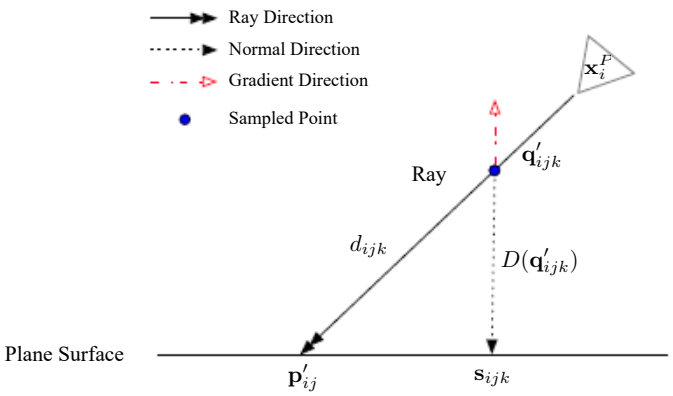


FIGURE 5.2: **Example of ray-surface-space consistency constraint in TSDF field.** The distance between $\mathbf{q}'_{ijk}$ and $\mathbf{s}_{ijk}$ can be obtained by $D(\mathbf{q}'_{ijk})$ in TSDF field. The ray direction in the global coordinate is $\mathbf{R}_i\mathbf{n}_{ij}$.

Substituting (5.9), (5.13) and (5.11) in (5.16), we obtain

$$\begin{aligned} f^C(\mathbf{x}) &= \sum_{i=0}^{n}\sum_{j}\sum_{k\in\Omega_C^i}\left\|\mathbf{n}_s{}^\top\left((\mathbf{R}_i\mathbf{q}_{ijk}+\mathbf{t}_i)-D(\mathbf{q}'_{ijk})\nabla D(\mathbf{q}'_{ijk})-(\mathbf{R}_i\mathbf{p}_{ij}+\mathbf{t}_i)\right)\right\|^2 \\ &= \sum_{i=0}^{n}\sum_{j}\sum_{k\in\Omega_C^i}\left\|\mathbf{n}_s{}^\top\left((\mathbf{R}_i(\mathbf{q}_{ijk}-\mathbf{p}_{ij})-D(\mathbf{q}'_{ijk})\nabla D(\mathbf{q}'_{ijk})\right)\right\|^2. \end{aligned} \tag{5.17}$$

From **Lemma 2**, $\mathbf{q}'_{ijk}$ and its associated closest surface point $\mathbf{s}_{ijk}$ share the same gradient:

$$\nabla D(\mathbf{q}'_{ijk}) = \nabla D(\mathbf{s}_{ijk}) = \mathbf{n}_{s_{ijk}}. \tag{5.18}$$

Substituting (5.15) and (5.18) into (5.17) yields the analytic form of the ray–surface–space consistency constraint:

$$f^C(\mathbf{x}) = \sum_{i=0}^{n}\sum_{j}\sum_{k\in\Omega_C^i}\left\|d_{ijk}\mathbf{n}_s^\top\mathbf{R}_i\mathbf{n}_{ij}-D(\mathbf{q}'_{ijk})\right\|^2, \tag{5.19}$$

where the local signed distance between $\mathbf{q}_{ijk}$ and $\mathbf{p}_{ij}$ is

$$d_{ijk} = \text{sign}\left(\|\mathbf{p}_{ij}\|-\|\mathbf{q}_{ijk}\|\right)\|\mathbf{p}_{ij}-\mathbf{q}_{ijk}\|, \tag{5.20}$$

and the j-th ray direction in the i-th local frame is

$$\mathbf{n}_{ij} = \frac{\mathbf{p}_{ij}}{\|\mathbf{p}_{ij}\|}. \tag{5.21}$$

As illustrated in Figure 5.2, (5.19) transforms the geometric relationship between $\mathbf{p}'_{ij}$, $\mathbf{s}_{ijk}$, and $\mathbf{q}'_{ijk}$ into a constraint linking the robot pose and the TSDF field within both the positive and negative truncated ray-tracing regions. This term complements the surface and space terms by incorporating richer geometric dependencies from the full ray structure.

4) Normalization Constraint Term: In the TSDF field, some voxels may not receive direct observations, which can lead to inconsistencies or drifting values in unobserved regions. To regularize these areas and maintain coherence across the TSDF space, we introduce a normalization constraint based on a discrete Laplacian operator:

$$f^N(\mathbf{x}) = \sum_{c \in \Omega_{tsdf}} \left\| \phi(\mathcal{N}(\mathbf{m}))D(\mathbf{m}_c) - \sum_{l \in \mathcal{N}(\mathbf{m})} D(\mathbf{m}_l) \right\|^2, \tag{5.22}$$

where $\mathcal{N}(\mathbf{m})$ denotes the set of neighboring voxels of $\mathbf{m}_c$ along the $\pm x$, $\pm y$, and $\pm z$ directions, and $\phi(\mathcal{N}(\mathbf{m}))$ is the cardinality of this set.

This Laplacian-based normalization encourages each voxel to remain consistent with the average of its neighbors and serves as a first-order smoothness prior on the TSDF field. It provides regularization in regions lacking measurements and complements the geometric constraints used elsewhere in the formulation.

5) Odometry Term: When available, the odometry measurements $\mathbb{O} = \{\mathbf{o}_i\}_{1 \le i \le n}$ provide relative motion information between consecutive robot poses. Each measurement is represented as $\mathbf{o}_i = [\mathbf{o}_i^t, \boldsymbol{\Delta}_{\xi_i}]$, where $\mathbf{o}_i^t \in \mathbb{R}^3$ is the relative translation and $\boldsymbol{\Delta}_{\xi_i} \in \mathbb{R}^3$ is the relative rotation expressed as a rotation vector in the axis–angle minimal parametrization of $\mathfrak{so}(3)$).

The odometry residual from robot pose $\mathbf{x}_{i-1}^P$ to pose $\mathbf{x}_i^P$ is defined as

$$f^O(\mathbf{x}) = \sum_{i=1}^{n} \left\| \begin{bmatrix} \mathbf{o}_i^t - \mathbf{R}_{i-1}(\mathbf{t}_i - \mathbf{t}_{i-1}) \\ \log\left(\exp(-\xi_i^\wedge)\exp(\xi_{i+1}^\wedge)\right) - \boldsymbol{\Delta}\xi_i \end{bmatrix} \right\|_{\boldsymbol{\Sigma}_{\mathbf{O_i}}^{-1}}, \tag{5.23}$$

in which $\mathbf{\Sigma}_{O_i}$ is the covariance matrix representing the uncertainty of the odometry measurement $\mathbf{o}_i$, and $\log(\cdot)$ denotes the logarithm map from SO(3) to $\mathfrak{so}(3)$.

5.2 Methodology

In this section, we present the complete methodology for solving the proposed problem. The overall procedure is summarized in Algorithm 6, and we now detail each component of the algorithm.

5.2.1 Ray-Surface-Space Consistency Filtering

To solve the proposed NLLS problem in (5.8), a prerequisite step is to determine whether each paired surface point satisfies the ray–surface–space consistency condition. Pairs meeting this criterion constitute the index set Ω_C^i, which is subsequently used in defining the ray–surface–space consistency term (5.19).

Equation (5.16) assumes that the two paired surface points $\{\mathbf{p}'_{ij}, \mathbf{s}_{ijk}\}$ correspond to locally consistent surface patches in the TSDF field. However, in complex environments, the nearest surface point $\mathbf{s}_{ijk}$ retrieved from an arbitrary sampled point within the ray-traced space may instead lie on a surface patch that is not mutually consistent with the one associated with the corresponding hit point $\mathbf{p}'_{ij}$. The possible relationships between these two points are illustrated in Figure 5.3. To address this issue, we introduce a ray-surface-space consistency filtering step to exclude any point pairs $\mathbf{p}'_{ij}, \mathbf{s}_{ijk}$ that do not meet the required consistency conditions before constructing the constraint term in (5.16). Specifically, two criteria are applied:

1) Normal Consistency: The normals (or gradients) of the two surface points $\mathbf{s}_{ijk}$ and $\mathbf{p}'_{ij}$ must be sufficiently similar to indicate that they belong to locally consistent surface patches. If the angular difference between the surface normals is below a threshold, the pair is considered consistent, i.e., $\|\mathbf{n}_{s_{ijk}} - \mathbf{n}_{p'_{ij}}\| = \|\nabla D(\mathbf{s}_{ijk}) - \nabla D(\mathbf{p}'_{ij})\| < \tau_{norm}$.

2) Line–Normal Orthogonality: When the normals of two surface points are consistent, we further check whether the projection of the line connecting these two points, $\mathbf{p}'_{ij} - \mathbf{s}_{ijk}$,

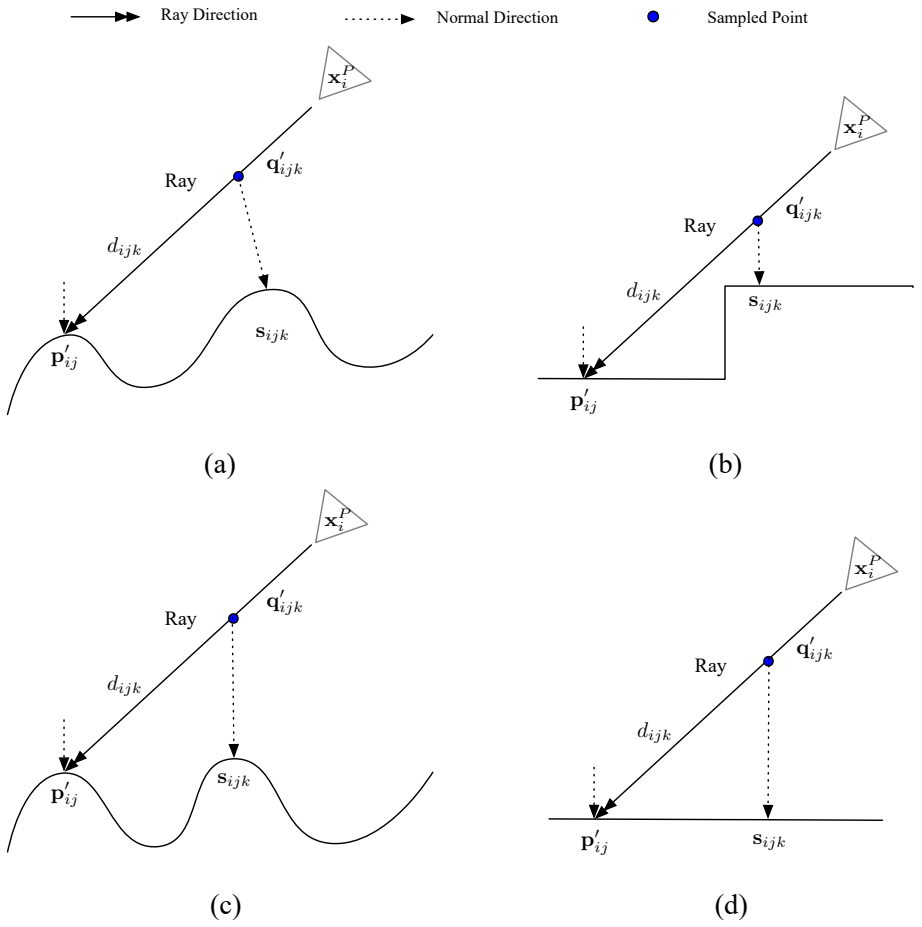


FIGURE 5.3: **Different cases of ray-surface-space consistency check.** Case (a) is rejected based on the difference in surface normals. Case (b) is rejected using the line–normal orthogonality check. Case (c) satisfies the ray–surface–space consistency constraint because the two points belong to locally consistent surface patches. Case (d) satisfies the ray–surface–space consistency constraint because the two points are located on the same plane.

onto the mean normal direction $\bar{\mathbf{n}}_{sp}$ is smaller than a threshold, i.e., $||\bar{\mathbf{n}}_{sp}^{\top}(\mathbf{p}'_{ij} - \mathbf{s}_{ijk})|| < \tau_{orth}$.

As illustrated in Figure 5.3, filter condition 1) rejects case (a), and filter condition 2) rejects case (b). In contrast, neither condition affects cases (c) and (d), both of which remain valid for the ray–surface–space consistency constraint term in (5.19).

Only the point pairs that satisfy both conditions are retained for the construction of the ray–surface–space consistency constraint term in (5.16). This filtering process corresponds to Line 8 in Algorithm 6.

5.2.2 Outlier Rejection for Ray–Surface–Space Consistency

The associations between $\mathbf{q}'_{ijk}$ and $\mathbf{s}_{ijk}$, and the corresponding ray–surface–space consistency, hold reliably when the TSDF field is accurate. However, this assumption may break down in practice, particularly when the optimization starts from non-ideal initial guesses. In such cases, inaccuracies in the TSDF field can lead to erroneous associations between $\mathbf{q}'_{ijk}$ and $\mathbf{s}_{ijk}$. Moreover, incorrect gradient evaluations at $\mathbf{s}_{ijk}$ and $\mathbf{p}'_{ij}$ may cause invalid pairs to erroneously pass the consistency filter and enter the set Ω^i_C, ultimately degrading optimization performance. To address this issue, we introduce an outlier rejection strategy to remove unreliable associations and improve robustness. Specifically, we apply two proximity-based outlier rejection strategies:

1) Surface Proximity: The closest surface point $\mathbf{s}_{ijk}$ associated with the projected sampled point $\mathbf{q}'_{ijk}$, computed using (5.11), must lie sufficiently close to the zero-level surface. That is, the absolute value of its signed distance value should be below a threshold $|D(\mathbf{s}_{ijk})| < \tau_{surf}$.

2) Spatial Proximity: To mitigate the influence of inaccurately evaluated surface normals, we introduce a spatial proximity check to validate whether two points can reliably contribute to the ray–surface–space consistency constraint. Specifically, we examine whether the Euclidean distance between $\|\mathbf{s}_{ijk}-\mathbf{p}'_{ij}\|$ is consistent with the sampled distance d_{ijk} scaled by the sine of the angle between the normals $\bar{\mathbf{n}}_s$ and $\mathbf{R}_i\mathbf{n}_{ij}$, i.e.,

$$d_{ijk}||\bar{\mathbf{n}}_s \times (\mathbf{R}_i\mathbf{n}_{ij})|| - \|\mathbf{s}_{ijk} - \mathbf{p}'_{ij}\| < \tau_{dist}. \tag{5.24}$$

Here, $\bar{\mathbf{n}}_s$ denotes the averaged normal vector computed from the normals at $\mathbf{n}_{s_{ijk}}$ and $\mathbf{n}_{p'_{ij}}$ by

$$\bar{\mathbf{n}}_s = \frac{\mathbf{n}_{s_{ijk}} + \mathbf{n}_{p'_{ij}}}{\left\|\mathbf{n}_{s_{ijk}} + \mathbf{n}_{p'_{ij}}\right\|}. \tag{5.25}$$

The two proximity-based outlier rejection strategies prevent unreliable associations from being incorporated into the optimization. From the optimization perspective, this mechanism ensures that when the TSDF field is inaccurate, the other four constraint terms dominate and guide the optimization more effectively. As optimization proceeds, the outlier

Algorithm 6: Joint Optimization Algorithm

Params: Thresholds τ_d, τ_k, τ_Δ, τ_{norm}, τ_{orth}, τ_{dist}, and τ_{surf}, weights w_0 , w_S, w_C, w_N, and w_O

Input: Point clouds $\mathbb{P}$, odometry $\mathbb{O}$

Output: Optimal poses $\hat{\mathbf{x}}^P$ and map $\hat{\mathbf{x}}^D$

1 $\mathbb{Z} \leftarrow$ `Sampling(`$\mathbb{P}$, τ_d`)`
2 $\mathbf{x}^P(0) \leftarrow$ `Pose Initialization(`$\mathbb{O}$`)`
3 $D(\cdot)$, $\mathbf{x}^D(0) \leftarrow$ `Mapping` ($\mathbf{x}^P(0)$, $\mathbb{Z}$)
4 **for** $k = 0$; $k <= \tau_k$ & $\|\boldsymbol{\Delta}(k)\|^2 >= \tau_\Delta$; $k++$ **do**
5 **for** $i = 0$; $i <= n$; $i++$ **do**
6 $\{\mathbf{p}'_{ij}, \mathbf{q}'_{ijk}\} \leftarrow$`Projection` ($\mathbf{x}_i^P(0)$, $\mathbb{Z}_i$)
7 $\{\mathbf{p}'_{ij}, \mathbf{s}_{ijk}\} \leftarrow$`Association` ($D(\cdot)$, $\{\mathbf{p}'_{ij}, \mathbf{q}'_{ijk}\}$)
8 $\tilde{\Omega}_C^i \leftarrow$ `RSS Consistency Filtering` ($D(\cdot)$, $\{\mathbf{p}'_{ij}, \mathbf{s}_{ijk}\}$, τ_{norm}, τ_{orth})
9 $\Omega_C^i \leftarrow$ `Outlier Rejection` ($D(\cdot)$, $\tilde{\Omega}_C^i$, τ_{surf}, τ_{dist})
10 **end**
11 $\mathbf{J} \leftarrow$ `Jacobian Calculation` ($\mathbb{Z}$, $\mathbf{x}^P(k)$, $\mathbf{x}^D(k), \Omega_C$)
12 $\boldsymbol{\Delta}(k) \leftarrow$ `Delta Calculation` ($\mathbf{J}$, $F(\mathbf{x})$)
13 $\mathbf{x}(k+1) \leftarrow$ `Update` ($\mathbf{x}(k)$, $\boldsymbol{\Delta}(k)$)
14 **end**
15 $\hat{\mathbf{x}}^P \Leftarrow \mathbf{x}^P(k)$, $\hat{\mathbf{x}}^D \Leftarrow \mathbf{x}^D(k)$
16 `return(`$\hat{\mathbf{x}}^P$, $\hat{\mathbf{x}}^D$`)`

rejection for ray–surface–space consistency is applied iteratively, meaning that the associations in Ω_C^i are dynamically updated during iterations. Consequently, once the TSDF field becomes reliable enough to provide accurate gradient evaluations, the ray–surface–space consistency term contributes more significantly. Overall, the outlier rejection process enhances the robustness of the optimization. This rejection step corresponds to Line 9 in Algorithm 6.

5.2.3 Solution to the NLLS Formulation

In Section 5.1.5, we introduced the NLLS formulation for the joint poses and TSDF map optimization problem. In this section, we introduce the solution to the proposed NLLS in (5.8).

Each constraint term contributes a residual vector: $F^0(\mathbf{x})$, $F^S(\mathbf{x})$, $F^C(\mathbf{x})$, $F^N(\mathbf{x})$, and $F^O(\mathbf{x})$, corresponding to the surface, space, ray-surface-space consistency, normalization,

and odometry terms, respectively. We collect all residuals into a single stacked vector

$$F(\mathbf{x}) = \left[F^0(\mathbf{x})^\top,\, F^S(\mathbf{x})^\top,\, F^C(\mathbf{x})^\top,\, F^N(\mathbf{x})^\top,\, F^O(\mathbf{x})^\top\right]^\top, \tag{5.26}$$

and define a block-diagonal weight matrix

$$\mathbf{W} = \mathrm{diag}(\cdots, w_0, \cdots, w_S, \cdots, w_{CP}, \cdots, w_N, \cdots, w_O\boldsymbol{\Sigma}_{O_i}^{-1}, \cdots), \tag{5.27}$$

which incorporates the weights of each term, with the odometry term weighted by the inverse of its covariance $\boldsymbol{\Sigma}_{O_i}^{-1}$.

The NLLS problem in (5.8) can then be written compactly as

$$f(\mathbf{x}) = \|F(\mathbf{x})\|_{\mathbf{W}}^2 = F(\mathbf{x})^\top \mathbf{W} F(\mathbf{x}), \tag{5.28}$$

where $\|\cdot\|_{\mathbf{W}}$ denotes the weighted ℓ_2-norm.

A solution to (5.28) can be obtained iteratively by starting with an initial guess $\mathbf{x}(0)$ and updating with $\mathbf{x}(k+1) = \mathbf{x}(k) + \boldsymbol{\Delta}(k)$. The update vector $\boldsymbol{\Delta}(k) = \left[\boldsymbol{\Delta}^P(k)^\top, \boldsymbol{\Delta}^D(k)^\top\right]^\top$ is the solution to

$$\mathbf{J}^\top \mathbf{W} \mathbf{J} \boldsymbol{\Delta}(k) = -\mathbf{J}^\top \mathbf{W} F(\mathbf{x}(k)) \tag{5.29}$$

where $\mathbf{J}$ is the linear mapping represented by the Jacobian matrix $\partial F/\partial \mathbf{x}$ evaluated at $\mathbf{x}(k)$.

5.2.4 Joint Optimization Algorithm

The procedure for solving the proposed problem is outlined in Algorithm 6, where τ_k and τ_Δ denote the maximum number of iterations and the threshold for the incremental update $\boldsymbol{\Delta}$, respectively.

Unlike conventional plane-constraint-based methods, which require explicit feature extraction, plane segmentation, and data association before optimization, our method avoids extracting any geometric features and explicit data association. In each iteration, after updating the robot poses and the TSDF map, we perform ray–surface–space consistency

filtering. In this way, pairwise associations are implicitly constructed and iteratively refined within the optimization rather than fixed beforehand. Among the proposed constraints, the surface and space terms rely only on the general properties of the signed distance field and are valid throughout the entire TSDF space. The ray–surface–space consistency term further exploits locally consistent surface patches where the ray endpoint and the inferred surface point are geometrically compatible, but it still operates directly on the TSDF without explicit plane fitting.

Because the surface and space constraints remain effective even when locally planar regions are sparse or noisy, the optimization problem stays well constrained, and the ray–surface–space consistency term gradually contributes more as the TSDF becomes more accurate. This design makes the proposed method robust to noisy initialization and avoids degradation caused by incorrect feature extraction or unreliable plane fitting.

5.3 Experimental Results

In this section, we evaluate the proposed algorithm presented in this chapter, hereafter SDF-BA, on two publicly available 3D LiDAR datasets and compare its performance with state-of-the-art methods, including LiDAR odometry method, a plane-feature-based BA method, and the OGM-based joint optimization approach introduced in Chapter 3. For trajectory accuracy, we employ ATE to assess 3-DoF performance and Absolute Pose Error (APE) to assess 6-DoF pose accuracy. All results are aligned with ground truth and evaluated using the EVO toolkit [126].

5.3.1 Datasets

We evaluate the proposed method on two real-world datasets, The Newer College Dataset [128] and Arche Dataset [10], which are also used for evaluation in Chapter 3 and Chapter 4. More details were provided in Section 3.5.2.2.

Since the proposed SDF-BA is designed as a local BA approach, each dataset is divided into multiple short subsequences, with each subsequence containing approximately 100

to 150 frames. It is worth noting that, due to poor GPS signal in certain segments of the Arche Dataset, some frames without ground truth are excluded from the evaluation subsequences.

5.3.2 State-of-the-art Methods Compared

In all experiments, the odometry provided by each dataset is used for initialization. To evaluate the proposed SDF-BA, we compare it against state-of-the-art methods: BALM2 [115], which represents the current best performance among plane-feature-based BA methods, and Occupancy-SLAM [16] [1], which is an OGM-based joint optimization approach. We also include comparisons with the odometry baselines, FAST-LIO2 [127] for The Newer College Dataset and ROVIO [129] for the Arche dataset. Both FAST-LIO2 and ROVIO leverage LiDAR-IMU or visual-inertial fusion, while BALM2, Occupancy-SLAM, and SDF-BA rely solely on LiDAR data. Therefore, in normal-scale environments, the comparison between LiDAR-only BA-based methods and LIO/VIO systems remains meaningful.

5.3.3 Pose Accuracy

We first evaluate the pose accuracy of SDF-BA against state-of-the-art approaches using The Newer College Dataset, which provides accurate 6-DoF ground truth for robot poses. The ATE results for both translation and rotation are presented in Table 5.1. As shown, SDF-BA consistently ranks first or second across nearly all metrics, and it achieves the lowest average translation error and the second-lowest average rotation error across all subsequences. The results demonstrate the effectiveness of SDF-BA in pose estimation. While Occupancy-SLAM achieves the best average rotation accuracy, it jointly optimizes robot poses and the entire OGM, which incurs significantly higher memory and computational costs. In contrast, SDF-BA jointly optimizes only the robot poses and the TSDF map. Under the same map resolution, using a TSDF instead of an OGM reduces the size of the map component in the state vector by several orders of magnitude—approximately the cube of a factor of a few dozen—which leads to significantly lower memory consumption and faster computation.

[1] 3D mode of the presented approach in Chapter 3.

Compared with BALM2, which explicitly extracts planar features and establishes plane relationships before optimization, SDF-BA does not rely on explicit feature extraction or fixed coplanar associations. BALM2 requires plane segmentation and data association as preprocessing steps. In contrast, SDF-BA implicitly identifies and refines geometric relationships through the ray–surface–space consistency mechanism during optimization. This dynamic, TSDF-driven association process allows geometric constraints to be continuously updated as the map improves, contributing to the higher accuracy achieved by SDF-BA.

It is also worth noting that FAST-LIO2 achieves the best rotation performance on several subsequences and ranks second overall in both translation and rotation accuracy. Although FAST-LIO2 is an odometry method, it fuses IMU and LiDAR measurements, enabling highly accurate short-term motion estimation, particularly for rotational components.

In addition to the evaluation on The Newer College Dataset, we further assess the trajectory accuracy of all compared methods on the Arche Dataset. Owing to significant vibrations during MAV flight, the odometry in this dataset is relatively noisy, making it a suitable benchmark for testing the robustness of SLAM algorithms. The results, presented in Table 5.2, show that SDF-BA achieves the second-best average performance in both MAE and RMSE metrics, closely following Occupancy-SLAM. Moreover, SDF-BA substantially outperforms both BALM2 and ROVIO, demonstrating strong robustness under challenging conditions.

BALM2 depends on explicit planar feature extraction, plane fitting, and fixed data association before optimization. As a result, its performance is highly sensitive to initialization quality. This limitation is evident in Seq. 18, Seq. 19, and Seq. 20, where BALM2 performs significantly worse than the raw odometry, indicating failures in reliable plane detection and association when the initialization is noisy. In contrast, SDF-BA does not require explicit geometric feature extraction and therefore maintains higher robustness and accuracy on this vibration-prone dataset.

TABLE 5.1: Absolute Pose Error (MAE/RMSE, Translation in Meters, Rotation in Radians) Evaluated by The Newer College Dataset for Different 3D Methods.

Seq.	ID		FAST-LIO2	BALM2	Occupancy-SLAM	SDF-BA
0	1	Trans	1.027/1.096	0.073/0.077	0.055/0.072	**0.028**/**0.040**
		Rot	1.073/1.074	0.796/0.796	**0.767**/**0.767**	0.784/0.784
	2	Trans	0.052/0.059	0.054/0.061	0.055/0.064	**0.051**/**0.058**
		Rot	**0.791**/**0.791**	**0.791**/**0.791**	**0.791**/**0.791**	**0.791**/**0.791**
	3	Trans	0.156/0.170	0.106/0.116	0.114/0.126	**0.098**/**0.109**
		Rot	0.788/0.788	**0.787**/**0.787**	0.788/0.788	**0.787**/**0.787**
1	1	Trans	0.158/0.173	0.130/0.139	**0.123**/**0.132**	0.127/0.137
		Rot	0.786/0.786	0.786/0.786	0.786/0.786	**0.785**/0.786
	2	Trans	0.071/**0.084**	**0.070**/**0.084**	0.074/0.088	0.071/**0.084**
		Rot	0.780/0.780	0.780/0.780	**0.778**/**0.779**	0.779/0.780
	3	Trans	0.077/**0.092**	0.079/0.094	0.085/0.100	**0.076**/**0.092**
		Rot	**0.782**/**0.782**	**0.782**/**0.782**	0.783/0.784	**0.782**/**0.782**
2	1	Trans	0.116/0.124	0.103/0.111	**0.081**/**0.087**	0.091/0.098
		Rot	**0.785**/**0.786**	0.787/0.787	**0.785**/**0.786**	0.787/0.787
	2	Trans	0.073/0.081	**0.060**/**0.064**	0.064/0.070	0.068/0.072
		Rot	0.789/0.789	0.790/0.790	**0.787**/**0.787**	**0.787**/**0.787**
	3	Trans	0.063/0.070	0.064/0.071	**0.062**/**0.068**	0.066/0.074
		Rot	0.798/0.798	**0.782**/0.783	0.784/0.784	**0.782**/**0.782**
3	1	Trans	0.117/0.131	0.069/0.080	**0.062**/**0.069**	0.064/0.071
		Rot	0.790/0.791	**0.786**/**0.787**	0.787/**0.787**	0.788/0.788
	2	Trans	0.080/0.094	0.084/0.093	**0.058**/**0.064**	0.073/0.081
		Rot	**0.787**/**0.788**	0.790/0.791	0.791/0.792	0.788/0.789
	3	Trans	0.095/0.116	0.097/0.117	0.113/0.134	**0.090**/**0.113**
		Rot	**0.786**/**0.786**	**0.786**/0.787	**0.786**/**0.786**	**0.786**/**0.786**
4	1	Trans	0.408/0.467	0.140/0.157	**0.080**/**0.096**	0.085/**0.096**
		Rot	**0.687**/**0.688**	0.753/0.754	0.769/0.770	0.771/0.772
	2	Trans	0.096/0.127	0.072/0.087	0.075/0.096	**0.061**/**0.075**
		Rot	**0.786**/0.787	0.787/0.787	**0.786**/**0.786**	**0.786**/**0.786**
	3	Trans	0.079/0.092	0.078/0.091	**0.068**/**0.080**	0.073/0.085
		Rot	0.782/0.783	0.782/0.782	**0.781**/**0.781**	**0.781**/0.782
Average		Trans	0.178/0.198	0.085/0.096	0.078 / 0.090	**0.075** / **0.086**
		Rot	0.799/0.800	0.784 / 0.785	**0.783** / **0.784**	0.784 / 0.785

Bold indicates the best performance and underline indicates the second best performance.

TABLE 5.2: Absolute Trajectory Error (MAE/RMSE, Meters) Evaluated by Arche Dataset for Different 3D Methods.

Seq.	ROVIO	BALM2	Occupancy-SLAM	SDF-BA
0	0.034/0.056	0.019/0.026	0.016/0.021	**0.012**/**0.016**
1	0.077/0.079	0.088/0.090	**0.069**/**0.071**	0.076/0.078
2	0.132/0.146	0.106/0.129	0.046/0.055	**0.033**/**0.038**
3	0.115/0.135	0.147/0.157	0.055/0.061	**0.037**/**0.040**
4	0.192/0.214	0.127/0.144	0.055/**0.060**	**0.053**/0.086
5	0.314/0.364	0.345/0.369	0.038/0.044	**0.026**/**0.042**
6	0.347/0.393	0.222/0.318	**0.058**/**0.110**	0.111/0.207
7	0.063/0.072	*/*	0.034/0.037	**0.015**/**0.016**
8	0.737/0.835	2.437/4.738	**0.036**/**0.040**	0.041/0.044
9	0.300/0.332	0.127/0.134	**0.040**/**0.046**	0.042/0.053
10	0.255/0.294	0.117/0.135	0.066/0.074	**0.056**/**0.066**
11	0.153/0.188	0.160/0.194	**0.118**/**0.122**	0.133/0.141
12	0.306/0.337	0.286/0.339	0.128/0.139	**0.099**/**0.123**
13	1.045/1.182	1.440/1.503	0.078/**0.085**	**0.073**/0.095
14	0.786/0.884	0.427/0.456	**0.117**/**0.123**	0.145/0.158
15	0.520/0.545	0.698/0.760	0.098/0.127	**0.070**/**0.081**
16	0.140/0.161	0.400/0.542	0.059/0.064	**0.043**/**0.048**
17	0.619/0.741	1.299/1.507	**0.107**/**0.127**	0.115/0.151
18	0.567/0.638	2.045/2.385	**0.037**/**0.042**	0.045/0.051
19	0.960/1.095	2.093/2.419	**0.074**/**0.170**	0.133/0.176
20	0.867/0.987	1.554/1.658	0.051/0.066	**0.047**/**0.053**
21	0.360/0.412	0.320/0.363	**0.069**/0.127	0.070/**0.105**
22	0.088/0.098	*/*	**0.077**/0.095	0.078/**0.090**
Average	0.390/0.443	0.689/0.875	**0.066**/**0.083**	0.068/0.085

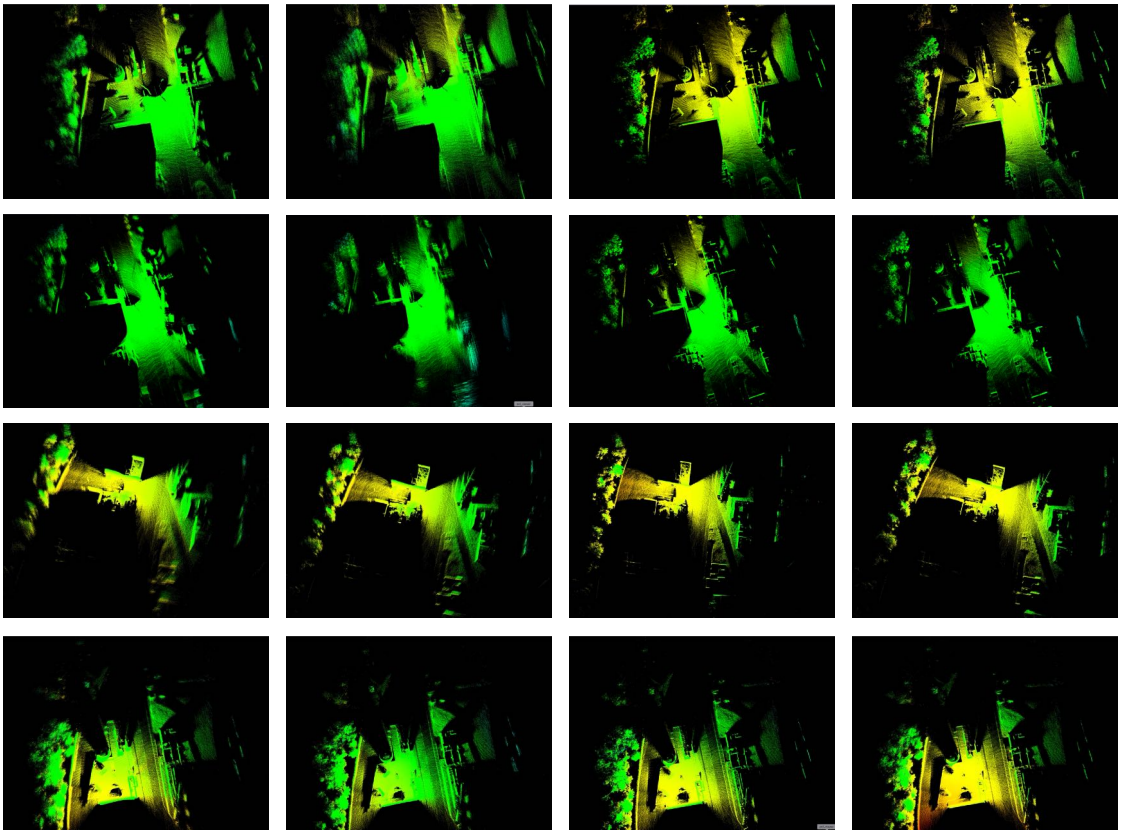
* indicates cases where the method failed to produce an optimized result. These instances are excluded from the average calculation.

Bold indicates the best performance and underline indicates the second best performance.

5.3.4 Comparisons of Maps

Since the Arche Dataset provides ground truth only for the x–y positions and does not include a ground truth map, we qualitatively evaluate the mapping performance by visualizing the maps generated from the optimized poses of BALM2, Occupancy-SLAM, and SDF-BA. For reference, we also include the map obtained using ROVIO odometry.

Because the compared methods rely on different map representations—SDF-BA uses a TSDF, Occupancy-SLAM uses an OGM, BALM2 extracts planar features from point



(a) ROVIO (b) BALM2 (c) Occupancy-SLAM (d) SDF-BA

FIGURE 5.4: **Comparisons of point cloud maps between different approaches on Arche Dataset.**

clouds, and ROVIO directly uses raw point clouds—we visualize both point cloud maps and TSDF maps to enable a fair and comprehensive comparison. The TSDF maps for ROVIO, Occupancy-SLAM, and BALM2 are generated using the same TSDF mapping procedure as in SDF-BA, based on their respective optimized poses, and are subsequently converted into meshes via the Marching Cubes algorithm for visualization.

The point cloud maps generated by the four methods are shown in Figure 5.4. The first two rows illustrate the degraded results from BALM2 on Seq.19 and Seq.20. In these cases, BALM2 performs noticeably worse than the odometry baseline (ROVIO), indicating that incorrect planar features were extracted, which in turn caused the optimization to diverge. In contrast, SDF-BA remains robust under the same noisy initialization, producing maps that are significantly more accurate than those obtained from both odometry and BALM2.

The last two rows show sequences where BALM2 does not exhibit degradation. Even in

(a) ROVIO (b) BALM2 (c) Occupancy-SLAM (d) SDF-BA

FIGURE 5.5: **Comparisons of mesh maps generated from TSDF between different approaches on Arche Dataset.**

these more favorable cases, SDF-BA still generates higher-quality maps than both BALM2 and odometry, further demonstrating its effectiveness. Compared with Occupancy-SLAM, SDF-BA delivers comparable map quality while being more computationally efficient, owing to the substantially smaller map component in its state vector.

Figure 5.5 shows the mesh reconstructions obtained from the optimized TSDF at a resolution of 0.2 meters. Across all sequences, SDF-BA consistently produces clearer and more accurate reconstructions than both ROVIO and BALM2, and achieves map quality on par with Occupancy-SLAM. These results further validate the robustness and effectiveness of the proposed method.

5.3.5 Ablation Study

To validate the effectiveness of each component in the proposed SDF-BA framework, we conduct an ablation study by systematically enabling or disabling specific constraint terms in the optimization. Three key components in the proposed formulation, the surface constraint, the space constraint, and the ray-surface-space consistency constraint, each leveraging different properties of the TSDF field. To eliminate the influence of external

TABLE 5.3: Absolute Pose Error (MAE/RMSE, Translation in Meters, Rotation in Radians) for Different Constraint Combinations on Seq. 0-ID1 of The Newer College Dataset.

Method	Seq0-ID1
surface + normalization (Trans)	1.027/1.096
surface + normalization (Rot)	1.073/1.074
RSS + normalization (Trans)	0.052/0.125
RSS + normalization (Rot)	0.784/0.784
surface + space + normalization (Trans)	1.027/1.095
surface + space + normalization (Rot)	1.073/1.073
surface + RSS + normalization (Trans)	0.031/0.057
surface + RSS +normalization (Rot)	0.787/0.787
space + RSS + normalization(Trans)	0.033/0.055
space + RSS + normalization(Rot)	0.783/0.783
RSS only (Trans)	0.052/0.125
RSS only (Rot)	0.784/0.784
surface + RSS (Trans)	0.431/0.502
surface + RSS (Rot)	0.955/0.956
space + RSS (Trans)	*/*
space + RSS (Rot)	*/*
full (Trans)	0.028/0.047
full (Rot)	0.780/0.781

* indicates cases where the configuration failed or does not converge.

priors and ensure that the evaluation purely reflects the effect of the proposed constraint terms, the odometry term is deliberately excluded in all configurations throughout this ablation study.

The evaluation is performed on ID 1 of Seq. 0 from The Newer College Dataset, representing a structured environment with high-quality data, and Seq. 12 from the Arche Dataset, containing more unstructured, non-planar scenes. This selection allows us to assess the performance of different constraints under varied environmental conditions.

All configurations use identical weight settings for consistency. The ablation study first includes the following configurations: 1) using surface and normalization constraint terms (denoted as surface constraint + normalization), 2) using ray-surface-space consistency and normalization constraint terms (RSS + normalization), 3) using surface, space and

normalization constraint terms (surface + space + normalization), 4) using surface, ray-surface-space consistency and normalization constraint terms (surface + RSS + normalization), 5) using space, ray-surface-space consistency and normalization constraint terms: (space + RSS + normalization), 6) only using the ray-surface-space consistency constraint term (RSS only), 7) using surface constraint term and ray-surface-space consistency constraint term (surface + RSS), and 8) using the space constraint term and ray-surface-space consistency constraint term (space + RSS). These configurations are compared against the full formulation to highlight the individual impact and mutual complementarity of each constraint component.

The results from ID 1 of Seq. 0 of The Newer College Dataset, presented in Table 5.3, demonstrate the significant role played by the ray-surface-space consistency constraint term. Specifically, configurations involving the ray-surface-space consistency constraint substantially outperformed those without it. The configuration "RSS + normalization" is significantly better than configurations relying only on surface-related constraints. These results align with the structured nature of The Newer College environment, which contains many high-quality planar surfaces. In addition, when comparing the "RSS + normalization" configuration with the proposed full method, it not only yields lower accuracy but also exhibits slower convergence. Specifically, we observe that the number of iterations required for convergence using only the ray-surface-space consistency and normalization constraint terms is approximately three times higher than that of the full formulation. This indicates that, especially under poor initialization, the inclusion of the surface and space constraint terms plays a crucial role in accelerating the convergence of the optimization process.

Removing the normalization term shows that "RSS only" is still effective, maintaining the same performance. However, removing normalization from the "surface + RSS" configuration significantly degrades performance, and the "space + RSS" configuration fails to converge, indicating the normalization term's critical role when using combined constraints. These results indicate that when employing the RSS constraint term, it is essential to maintain consistency across the entire TSDF space. Without this consistency, conflicts may arise between constraint terms, ultimately degrading the optimization performance.

TABLE 5.4: Absolute Trajectory Error (MAE/RMSE, Translation in Meters, Rotation in Radians) for Different Constraint Combinations on Seq. 12 of Arche Dataset.

Method	Seq.12
surface + normalization	0.306/0.337
RSS + normalization	0.123/0.157
surface + space + normalization	0.301/0.339
surface + RSS + normalization	0.108/0.139
space + RSS + normalization	0.122/0.163
RSS only	*/*
surface + RSS	0.103/0.125
space + RSS	*/*
full	0.099/0.123

* indicates cases where the configuration failed or does not converge.

This highlights the necessity of incorporating the other three terms in the proposed formulation to ensure robust and accurate results.

Table 5.4 shows results for Seq. 12 of the Arche Dataset. The configurations involving the ray-surface-space consistency term significantly reduce errors compared to the surface and space-only configurations. Specifically, the "RSS + normalization" configuration is far superior to the "surface + normalization" configuration. Combining the surface constraint with the ray-surface-space consistency constraint further improves performance slightly. Similar to The Newer College results, configurations relying solely on the ray-surface-space consistency constraint or space and ray-surface-space consistency constraints without normalization fail or do not converge, highlighting the importance of the normalization term. However, it should be noted that, unlike the results on The Newer College Dataset, using only the ray-surface-space consistency constraint term leads to optimization failure on Arche Dataset. This highlights the effectiveness of the other constraint terms in preventing optimization failure when the initialization does not provide sufficiently reliable geometric information. These findings demonstrate that the additional terms enhance the robustness of the proposed algorithm.

For both The Newer College and Arche datasets, the full formulation consistently achieves the lowest translation errors, demonstrating the complementary strengths of all integrated constraint terms. Although we present only one representative sequence from each dataset,

we note that the same trends hold across all sequences: the full formulation always performs best, while the relative contributions of the individual constraint terms remain consistent.

In summary, the ablation study clearly reveals the pivotal role of the ray-surface-space consistency constraint in achieving robust and accurate optimization. It also demonstrates that integrating multiple constraints, along with the critical normalization term, consistently provides the best performance across structured and unstructured environments.

5.4 Chapter Summary

In this chapter, we formulate the joint optimization of robot poses and a TSDF as a NLLS problem, where the state vector includes all robot poses and the signed distance values at the cell vertices of the TSDF map. Unlike existing SDF-based SLAM approaches that focus only on the zero-level set of the TSDF and optimize only the poses, the proposed SDF-BA encodes both geometric and field-level constraints across the entire TSDF space, jointly optimizing poses and all TSDF cells. Specifically, we formulate surface, space, and ray-surface-space consistency constraints within the NLLS framework to enable more accurate optimization compared to state-of-the-art methods.

Benefiting from the properties of the TSDF representation and an implicit, iteration-dependent data association mechanism, SDF-BA demonstrates greater robustness than existing plane-feature-based BA methods. Furthermore, since the formulation of SDF-BA models only the relevant region of the TSDF space, the size of the map component in the state vector is significantly smaller than that of OGM-based joint optimization schemes, resulting in much lower memory consumption and substantially faster computation.

Experimental evaluations on two publicly available 3D LiDAR datasets confirm the superior accuracy and efficiency of the proposed method.

Chapter 6

Conclusions and Future Work

In this thesis, we investigate the joint optimization of robot poses and non-feature-based maps. While various types of non-feature representations exist, our focus is on OGMs and SDFs. We demonstrate that jointly optimizing both robot poses and non-feature maps (OGMs and SDFs) leads to more accurate pose and map estimates compared to approaches that perform pose estimation first followed by non-feature mapping. Furthermore, our results show that by leveraging the inherent spatial properties of non-feature representations, the proposed methods can outperform traditional feature-based approaches, such as plane-feature-based techniques.

In the first study presented in Chapter 3, we propose an OGM-based joint optimization method in which both robot poses and the occupancy map are optimized simultaneously. This method introduces a variant of the GN algorithm to solve the newly formulated problem, enabling more accurate estimation of both robot trajectories and occupancy maps compared to existing state-of-the-art techniques. The approach is first developed for 2D scenarios, incorporating a multi-resolution optimization strategy to improve efficiency and accuracy. It is then extended to 3D settings, where it demonstrates superior performance over current 3D LiDAR-based SLAM methods. To the best of our knowledge, this is the first work that focuses on the joint optimization of robot poses and non-feature-based maps, in which the relationship between observations and the map is not formulated as position-to-position correspondences.

In the second study presented in Chapter 4, we propose an OGM-based submap joining algorithm that jointly optimizes both the coordinate frames of local submaps and the global OGM. The proposed formulation exhibits a special property: when solved using the GN method, the pose updates at each iteration are independent of the occupancy values in the global map. Leveraging this property, we design a pose-only GN method that is mathematically equivalent to the full GN approach, while significantly improving computational efficiency. This submap joining algorithm effectively extends the joint optimization framework introduced in Chapter 3 to large-scale SLAM problems. Initially developed in 2D scenarios, the method demonstrates superior performance in both accuracy and efficiency compared to existing state-of-the-art techniques. We further extend the algorithm to 3D environments, where it also achieves improved accuracy over current leading 3D SLAM approaches.

In addition to exploring the joint optimization of robot poses and OGMs, the third work presented in Chapter 5 investigates whether similar joint optimization can be extended to other types of non-feature-based representations. Specifically, this chapter focuses on the joint optimization of robot poses and SDFs. Unlike the OGM-based methods introduced in Chapters 3 and 4, which formulate the optimization problem by modeling relationships between observations and OGMs using occupancy values, this work leverages the geometric information inherently encoded in the SDF representation. The proposed framework exploits the fundamental properties of the TSDF field to define surface, space, and ray–surface–space consistency constraints within the volumetric space. Rather than relying on conventional SDF-based registration techniques, such as point-to-SDF or SDF-to-SDF matching, our method directly incorporates both geometric constraints and field-level information from the TSDF. This enables a richer and more integrated use of spatial structure. Furthermore, in contrast to traditional plane-feature-based BA approaches that depend on explicit planar extraction and pre-defined data association, our approach embeds such geometric relationships implicitly within the TSDF structure. This implicit and adaptive mechanism provides greater flexibility and robustness, particularly in environments where reliable planar features are scarce or difficult to extract.

Building on all the presented work, it becomes evident that jointly optimizing robot poses and non-feature-based representations yields more accurate results than the conventional

decoupled approach, where pose estimation is first performed using feature-based methods followed by non-feature-based mapping as a separate process. The decoupled strategy often fails to fully exploit the structural properties embedded in non-feature representations. Furthermore, compared to feature-based BA approaches, such as plane-feature-based BA, the proposed methods for joint optimization with non-feature maps demonstrate greater accuracy and robustness. One key advantage is that non-feature-based methods do not require explicit data association prior to optimization. Instead, they dynamically adjust the associations during the optimization process, increasing resilience to noise and initialization errors. In addition, the proposed methods leverage the rich spatial information inherently encoded in non-feature representations, such as occupancy probabilities indicating the likelihood of space being occupied, or signed distance fields that quantify the distance from any point in space to the nearest surface. These representations go beyond modeling obstacles alone. They provide continuous and differentiable fields that guide optimization more effectively. Moreover, the spatial representations are optimized jointly with poses, rather than being extracted and fixed beforehand, further enhancing the adaptability and performance of the system. These characteristics collectively contribute to the effectiveness of non-feature-based joint optimization frameworks in achieving accurate and robust SLAM performance.

Another important question concerns the differences among various non-feature map representations within the joint optimization framework. Since there exist numerous types of non-feature representations, each encoding the environment in fundamentally different ways, it is challenging to generalize a unified solution. Although this thesis focuses on leveraging OGMs and SDFs for the joint optimization of poses and non-feature maps, it does not exhaust all possible design choices.

However, by comparing the proposed OGM-based and SDF-based joint optimization approaches, a key difference lies in how observations are encoded to make them suitable for formulating the corresponding optimization problem. This process involves establishing the relationships between observations and non-feature maps through the robot poses. Such formulation challenges are a core reason why joint optimization with non-feature maps is more complex than traditional feature-based or direct BA methods. Furthermore, the intrinsic properties of different non-feature representations influence their suitability for

different tasks. For example, SDFs provide naturally continuous and smooth gradients, which are beneficial for geometric optimization, while OGMs offer better robustness to noise due to their probabilistic formulation. Therefore, different joint optimization strategies are better suited to different application scenarios. It is also worth exploring how other types of non-feature representations could be incorporated into joint optimization frameworks, and whether alternative formulations of OGM-based or SDF-based methods can lead to improved performance and broader applicability.

There remain several promising directions for future work:

- **Exploring other non-feature map representations in the joint optimization problem:** Emerging representations such as NeRF and Gaussian Splatting have shown strong potential in SLAM and downstream robotics tasks. It is worth investigating how their unique properties can be leveraged within a joint optimization framework to enhance the performance of non-feature-based SLAM systems.
- **Incorporating continuous map representations:** Continuous map representations may offer advantages in terms of optimization smoothness, memory efficiency, and computational speed. In our current OGM-based joint optimization approach, discrete occupancy grids are used, which introduce quantization artifacts that require additional smoothing. Future research could explore the use of continuous occupancy models within the joint optimization framework, particularly for large-scale 3D SLAM applications.
- **Developing more efficient TSDF storage and access structures:** The current TSDF representation in our joint optimization framework uses a dense grid, which is simple but memory-intensive and not ideal for large-scale environments. Future work may explore sparse and hierarchical volumetric data structures, such as OpenVDB [136], voxel hashing [44], or multi-resolution voxel grids [76], to store and access the TSDF more efficiently. Such representations can significantly reduce memory usage, accelerate queries during optimization, and enable scalable SDF-based joint optimization in large 3D environments.

Appendix A

Analytical Jacobian of Joint Optimization of Robot Trajectories and Occupancy Maps

The Jacobian $\mathbf{J}$ in (3.18) consists of four parts, i.e. the Jacobian of the observation term w.r.t. the robot poses $\mathbf{J}_P$ (See Appendix A.1), the Jacobian of the observation term w.r.t. the occupancy map $\mathbf{J}_M$ (See Appendix A.2), the Jacobian of the odometry term w.r.t. robot poses $\mathbf{J}_O$ (See Appendix A.3) and the Jacobian of the smoothing term w.r.t. the occupancy map $\mathbf{J}_S$ (See Appendix A.4). In addition, the difference in the calculation of Jacobians between Algorithm 1 and Algorithm 3 is shown in Appendix A.5.

A.1 Jacobian of the Observation Term w.r.t. Robot Poses

The Jacobian $\mathbf{J}_P$ of function $F_{ij}^Z(\mathbf{x})$ in the observation term w.r.t. the robot poses $\mathbf{x}_i^P$ can be calculated by the chain rule

$$\mathbf{J}_P = \frac{\partial F_{ij}^Z(\mathbf{x})}{\partial \mathbf{x}_i^P} = \frac{\partial F_{ij}^Z(\mathbf{x})}{\partial \mathbf{p}_{ij}} \cdot \frac{\partial \mathbf{p}_{ij}}{\partial \mathbf{x}_i^P} \tag{A.1}$$

in which $\frac{\partial \mathbf{p}_{ij}}{\partial \mathbf{x}_i^P}$ can be calculated as

$$\frac{\partial \mathbf{p}_{ij}}{\partial \mathbf{x}_i^P} = \begin{bmatrix} \frac{\partial \mathbf{p}_{ij}}{\partial \mathbf{t}_i} & \frac{\partial \mathbf{p}_{ij}}{\partial \theta_i} \end{bmatrix} = \frac{1}{s} \begin{bmatrix} \mathbf{E}_2 & \mathbf{R}_i' \mathbf{p}_{ij} \end{bmatrix}. \tag{A.2}$$

$\mathbf{R}_i'$ is the derivative of the rotation matrix $\mathbf{R}_i$ w.r.t. rotation angle θ_i and $\mathbf{E}_2$ means 2×2 identity matrix.

$\frac{\partial F_{ij}^Z(\mathbf{x})}{\partial \mathbf{p}_{ij}}$ can be calculated by

$$\frac{\partial F_{ij}^Z(\mathbf{x})}{\partial \mathbf{p}_{ij}} = \frac{1}{N(\mathbf{p}_{ij})} \frac{\partial M(\mathbf{p}_{ij})}{\partial \mathbf{p}_{ij}}. \tag{A.3}$$

Here $\frac{\partial M(\mathbf{p}_{ij})}{\partial \mathbf{p}_{ij}}$ can be considered as the gradient of the occupancy map at point $\mathbf{p}_{ij}$, which can be approximated by the bilinear interpolation of the gradients of the occupancy at the four adjacent cell vertices $\nabla M(\mathbf{m}_{wh}), \cdots, \nabla M(\mathbf{m}_{(w+1)(h+1)})$ around $\mathbf{p}_{ij}$ as

$$\frac{\partial M(\mathbf{p}_{ij})}{\partial \mathbf{p}_{ij}} = \begin{bmatrix} a_1 b_1 \\ a_0 b_1 \\ a_1 b_0 \\ a_0 b_0 \end{bmatrix}^\top \begin{bmatrix} \nabla M(\mathbf{m}_{wh}) \\ \nabla M(\mathbf{m}_{(w+1)h}) \\ \nabla M(\mathbf{m}_{w(h+1)}) \\ \nabla M(\mathbf{m}_{(w+1)(h+1)}) \end{bmatrix} \tag{A.4}$$

where the gradient of occupancy map $\mathbb{M}$ at all the cell vertices ∇M can be easily calculated from $\mathbf{x}^M$ in the state. The bilinear interpolation used in (A.4) is similar to the method in (3.1).

Here, we assume the robot poses $\mathbf{x}^P$ change slightly in each iteration, to reduce the computational complexity, the hit map $\mathbb{N}$ is considered as constant and recalculated using the current robot poses in each iteration. Thus, the derivative of $N(\mathbf{p}_{ij})$ is not calculated.

A.2 Jacobian of the Observation Term w.r.t. Occupancy Map

Based on (3.1), the Jacobian $\mathbf{J}_M$ of function $F_{ij}^Z(\mathbf{x})$ in the observation term w.r.t. the map part of state vector $\mathbf{x}^M$ can be calculated as

$$\begin{aligned}\mathbf{J}_M &= \frac{\partial F_{ij}^Z(\mathbf{x})}{\partial \left[M(\mathbf{m}_{wh}), \cdots, M(\mathbf{m}_{(w+1)(h+1)})\right]^\top} \\ &= \frac{1}{N(\mathbf{p}_{ij})} \frac{\partial M(\mathbf{p}_{ij})}{\partial \left[M(\mathbf{m}_{wh}), \cdots, M(\mathbf{m}_{(w+1)(h+1)})\right]^\top} \\ &= \frac{\left[a_1 b_1, a_0 b_1, a_1 b_0, a_0 b_0\right]}{N(\mathbf{p}_{ij})}\end{aligned} \tag{A.5}$$

where $\mathbf{m}_{wh}, \cdots, \mathbf{m}_{(w+1)(h+1)}$ are the four nearest cell vertices to $\mathbf{p}_{ij}$ in occupancy map $\mathbb{M}$, and a_0, a_1, b_0 and b_1 are defined in (3.1).

A.3 Jacobian of the Odometry Term

The Jacobian $\mathbf{J}_O$ of function $F_i^O(\mathbf{x})$ in the odometry term (3.14) is the partial derivative w.r.t. the robot poses $\mathbf{x}^P$ since it is not related to the occupancy map in the state vector $\mathbf{x}$. Therefore, the Jacobian $\mathbf{J}_O$ can be calculated as

$$\begin{aligned}\mathbf{J}_O &= \frac{\partial F_i^O(\mathbf{x})}{\partial \left[{\mathbf{x}_{i-1}^P}^\top, {\mathbf{x}_i^P}^\top\right]^\top} \\ &= \begin{bmatrix} \frac{\partial F_i^O(\mathbf{x})}{\partial \mathbf{t}_{i-1}} & \frac{\partial F_i^O(\mathbf{x})}{\partial \theta_{i-1}} & \frac{\partial F_i^O(\mathbf{x})}{\partial \mathbf{t}_i} & \frac{\partial F_i^O(\mathbf{x})}{\partial \theta_i} \end{bmatrix} \\ &= \begin{bmatrix} -\mathbf{R}_{i-1}^\top & (\mathbf{R}'_{i-1})^\top(\mathbf{t}_i - \mathbf{t}_{i-1}) & \mathbf{R}_{i-1}^\top & \mathbf{0}_2 \\ \mathbf{0}_2^\top & -1 & \mathbf{0}_2^\top & 1 \end{bmatrix}\end{aligned} \tag{A.6}$$

in which $\mathbf{0}_2$ means 2×1 zero vector.

A.4 Jacobian of the Smoothing Term

The Jacobian $\mathbf{J}_S$ of function $F^S(\mathbf{x})$ in the smoothing term is the derivative of (3.15) w.r.t. cell vertices of occupancy map $\mathbf{x}^M$ due to it is not related to the robot poses $\mathbf{x}^P$ in the state vector $\mathbf{x}$. It should be mentioned that $F^S(\mathbf{x})$ is linear w.r.t. $\mathbf{x}^M$

$$F^S(\mathbf{x}) = \mathbf{A}\left[M(\mathbf{m}_{00}), \cdots, M(\mathbf{m}_{c_w c_h})\right]^\top \tag{A.7}$$

where the $(2c_w c_h + c_w + c_h) \times ((c_w + 1)(c_h + 1))$ coefficient matrix $\mathbf{A}$ is sparse and with nonzero elements 1 or -1. An example of the coefficient matrix can be shown as

$$\mathbf{A} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}^\top & 1 & -1 & 0 & \mathbf{0}^\top & 0 & 0 & \mathbf{0}^\top \\ \mathbf{0}^\top & 1 & 0 & 0 & \mathbf{0}^\top & -1 & 0 & \mathbf{0}^\top \\ \mathbf{0}^\top & 0 & 1 & -1 & \mathbf{0}^\top & 0 & 0 & \mathbf{0}^\top \\ \mathbf{0}^\top & 0 & 1 & 0 & \mathbf{0}^\top & 0 & -1 & \mathbf{0}^\top \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{A.8}$$

Here $\mathbf{0}$ represents a zero vector with appropriate dimensions. Therefore, the Jacobian of the smoothing term can be calculated as

$$\mathbf{J}_S = \frac{\partial F^S(\mathbf{x})}{\partial \mathbf{x}^M} = \mathbf{A}. \tag{A.9}$$

Since $\mathbf{A}$ is constant, $\mathbf{J}_S$ can be pre-calculated and directly used in the optimization as shown in Algorithm 1.

A.5 Jacobians in the Second Stage of Multi-resolution Strategy for Optimization

In the second stage of the multi-resolution strategy (Algorithm 3), the Jacobians to be calculated are similar to those in Algorithm 1. A specific challenge arises in handling

the selected cell vertices adjacent to the dropped cell vertices in the high-resolution map, particularly when calculating Jacobians $\mathbf{J}_P$ and $\mathbf{J}_S$.

For Jacobian $\mathbf{J}_P$, partial derivatives w.r.t. all the cell vertices are required for (A.4). However, not all vertices are included in the state vector in the second stage, which makes it challenging to compute the partial derivatives w.r.t. some cell vertices because their surrounding nodes are discarded. From a semantic perspective, the discarded cell vertices have the same occupancy state as the edge nodes, which is why they are excluded. Consequently, the gradient of these edge vertices is expected to be close to zero. Based on this reasoning, we set the partial derivatives w.r.t. all edge cell vertices to 0 when they need to be calculated using (A.4).

For Jacobian $\mathbf{J}_S$, it can also be calculated using the same idea as (A.9). In the second stage of our multi-resolution strategy, (A.9) is reformulated as

$$\mathbf{J}_S = \frac{\partial F_s^S(\mathbf{x}^s)}{\partial \mathbf{x}^{sM}} = \mathbf{A}^s \tag{A.10}$$

where $\partial F_s^S(\mathbf{x}^s)$ is similar to (3.15), but only applies to vertices in $\mathbb{M}^s$. The coefficient matrix $\mathbf{A}^s$ has the same form as (A.8) but with dimension corresponding to the number of elements in $\mathbf{x}^{sM}$.

Bibliography

[1] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[2] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[3] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31 (5):1147–1163, 2015.

[4] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 4503–4508. IEEE, 2017.

[5] Ji Zhang, Sanjiv Singh, et al. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[6] Helen Oleynikova, Alexander Millane, Zachary Taylor, Enric Galceran, Juan Nieto, and Roland Siegwart. Signed distance fields: A natural representation for both mapping and planning. In *RSS 2016 workshop: geometry and beyond-representations, physics, and scene understanding for robotics*. University of Michigan, 2016.

[7] Stefan Kohlbrecher, Oskar von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable SLAM system with full 3D motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160, Kyoto, Japan, November 2011. IEEE. ISBN 978-1-61284-769-6 978-1-61284-770-2 978-1-61284-768-9. doi: 10.1109/SSRR.2011.6106777.

[8] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *Proceedings of 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

[9] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: science and systems*, volume 11, page 3. Rome, 2015.

[10] 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, 2019.

[11] Yiduo Wang, Nils Funk, Milad Ramezani, Sotiris Papatheodorou, Marija Popović, Marco Camurri, Stefan Leutenegger, and Maurice Fallon. Elastic and efficient lidar reconstruction for large-scale exploration tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5041. IEEE, 2021.

[12] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[13] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[14] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proceedings of 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160. IEEE, 2011.

[15] Liang Zhao, Yingyu Wang, and Shoudong Huang. Occupancy-SLAM: Simultaneously Optimizing Robot Poses and Continuous Occupancy Map. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi: 10.15607/RSS.2022.XVIII.003.

[16] Yingyu Wang, Liang Zhao, and Shoudong Huang. Occupancy-slam: An efficient and robust algorithm for simultaneously optimizing robot poses and occupancy map. *IEEE Transactions on Robotics*, 2025. doi: 10.1109/TRO.2025.3578227.

[17] Yingyu Wang, Liang Zhao, and Shoudong Huang. Grid-based submap joining: An efficient algorithm for simultaneously optimizing global occupancy map and local submap frames. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10121–10128. IEEE, 2024.

[18] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Proceedings of 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121. IEEE, 1985.

[19] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. In *Sensor Devices and Systems for Robotics*, pages 253–276. Springer, 1989.

[20] Alberto Elfes. *Occupancy grids: A probabilistic framework for robot perception and navigation*. Carnegie Mellon University, 1989.

[21] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[22] 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, 2013.

[23] Pierre Payeur, Patrick Hébert, Denis Laurendeau, and Clément M Gosselin. Probabilistic octree modeling of a 3d dynamic environment. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1289–1296. IEEE, 1997.

[24] Jonathan Fournier, Benoit Ricard, and Denis Laurendeau. Mapping and exploration of complex environments using persistent 3d model. In *Fourth Canadian Conference on Computer and Robot Vision (CRV'07)*, pages 403–410. IEEE, 2007.

[25] Kaustubh Pathak, Andreas Birk, Jann Poppinga, and Soren Schwertfeger. 3d forward sensor modeling and application to occupancy grid based sensor fusion. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2059–2064. IEEE, 2007.

[26] Sheraz Khan, Athanasios Dometios, Chris Verginis, Costas Tzafestas, Dirk Wollherr, and Martin Buss. Rmap: a rectangular cuboid approximation framework for 3d environment mapping. *Autonomous Robots*, 37(3):261–277, 2014.

[27] Alex Fisher, Ricardo Cannizzaro, Madeleine Cochrane, Chatura Nagahawatte, and Jennifer L Palmer. Colmap: A memory-efficient occupancy grid mapping framework. *Robotics and Autonomous Systems*, 142:103755, 2021.

[28] Victor Reijgwart, Cesar Cadena, Roland Siegwart, and Lionel Ott. Efficient volumetric mapping of multi-scale environments using wavelet-based compression. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.065.

[29] 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, 2018.

[30] Kumar Shaurya Shankar and Nathan Michael. Mrfmap: Online probabilistic 3d mapping using forward ray sensor models. *arXiv preprint arXiv:2006.03512*, 2020.

[31] Simon O'Callaghan, Fabio T Ramos, and Hugh Durrant-Whyte. Contextual occupancy maps using gaussian processes. In *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, pages 1054–1060. IEEE, 2009.

[32] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

[33] Soohwan Kim and Jonghyuk Kim. Building occupancy maps with a mixture of gaussian processes. In *Proceedings of 2012 IEEE International Conference on Robotics and Automation*, pages 4756–4761. IEEE, 2012.

[34] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots*, 42:273–290, 2018.

[35] 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, 2016.

[36] Ransalu Senanayake and Fabio Ramos. Bayesian hilbert maps for dynamic continuous occupancy mapping. In *Conference on Robot Learning*, pages 458–471. PMLR, 2017.

[37] Weiming Zhi, Lionel Ott, Ransalu Senanayake, and Fabio Ramos. Continuous occupancy map fusion with fast bayesian hilbert maps. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4111–4117. IEEE, 2019.

[38] Thai Duong, Michael Yip, and Nikolay Atanasov. Autonomous navigation in unknown environments with sparse bayesian kernel-based occupancy mapping. *IEEE Transactions on Robotics*, 38(6):3694–3712, 2022.

[39] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[40] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5379–5389, 2019.

[41] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9223–9232, 2023.

[42] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[43] Victor Reijgwart, Jens Behley, Teresa Vidal-Calleja, Helen Oleynikova, Lionel Ott, Cyrill Stachniss, and Ayoung Kim. *Dense Map Representation*. Cambridge University Press.

[44] 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. IEEE, 2017.

[45] Mayank Mittal, David Hoeller, Farbod Farshidian, Marco Hutter, and Animesh Garg. Articulated object interaction in unknown scenes with whole-body mobile manipulation. In *2022 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1647–1654. IEEE, 2022.

[46] Miroslava Slavcheva, Wadim Kehl, Nassir Navab, and Slobodan Ilic. Sdf-2-sdf registration for real-time 3d reconstruction from rgb-d data. *International Journal of Computer Vision*, 126(6):615–636, 2018.

[47] Tiancheng Li, Peter Walker, Danial Hammoud, Liang Zhao, and Shoudong Huang. Partial-to-full registration based on gradient-sdf for computer-assisted orthopedic surgery. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4520–4526. IEEE, 2025.

[48] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011.

[49] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[50] James A Sethian. A fast marching level set method for monotonically advancing fronts. *proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[51] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005.

[52] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[53] Stanley Osher, Ronald Fedkiw, and Krzysztof Piechor. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.*, 57(3):B15–B15, 2004.

[54] James A Sethian et al. *Level set methods and fast marching methods*, volume 98. Cambridge Cambridge UP, 1999.

[55] Boris Lau, Christoph Sprunk, and Wolfram Burgard. Improved updating of euclidean distance maps and voronoi diagrams. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 281–286. IEEE, 2010.

[56] 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. IEEE, 2019.

[57] 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, 2021.

[58] 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, 2022.

[59] 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*, 2022.

[60] Yulun Tian, Hanwen Cao, Sunghwan Kim, and Nikolay Atanasov. Miso: Multiresolution submap optimization for efficient globally consistent neural implicit reconstruction. In *Robotics: Science and Systems*, 2025.

[61] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000.

[62] Peter Biber and Wolfgang 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. IEEE, 2003.

[63] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[64] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4): 139–1, 2023.

[65] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *AAAI/IAAI*, 593598, 2002.

[66] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2432–2437. IEEE, 2005.

[67] Kurt Konolige, Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent. Efficient sparse pose adjustment for 2d mapping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–29. IEEE, 2010.

[68] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of field robotics*, 36(2):416–446, 2019.

[69] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[70] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, 1995.

[71] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and systems (RSS) conference 2013*, volume 9. Robotics: Science and Systems, 2013.

[72] Michael Strecke and Jorg Stuckler. Em-fusion: Dynamic object-level slam with probabilistic data association. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2019.

[73] Christiane Sommer, Lu Sang, David Schubert, and Daniel Cremers. Gradient-sdf: A semi-implicit surface representation for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6280–6289, 2022.

[74] Olaf Kähler, Victor Adrian Prisacariu, and David W. Murray. Real-time large-scale dense 3d reconstruction with loop closure. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, pages 500–516, 2016.

[75] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S Torr, and D. W. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. *IEEE Transactions on Visualization and Computer Graphics (Proceedings International Symposium on Mixed and Augmented Reality 2015*, 22(11), 2015.

[76] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul HJ Kelly, and Stefan Leutenegger. Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2): 1144–1151, 2018.

[77] Emanuele Vespa, Nils Funk, Paul HJ Kelly, and Stefan Leutenegger. Adaptive-resolution octree-based volumetric slam. In *2019 International Conference on 3D Vision (3DV)*, pages 654–662. IEEE, 2019.

[78] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40 (12-14):1510–1546, 2021.

[79] Jianyuan Ruan, Bo Li, Yibo Wang, and Yuxiang Sun. Slamesh: Real-time lidar simultaneous localization and meshing. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3546–3552. IEEE, 2023.

[80] Jiarong Lin, Chongjian Yuan, Yixi Cai, Haotian Li, Yunfan Ren, Yuying Zou, Xiaoping Hong, and Fu Zhang. Immesh: An immediate lidar localization and meshing framework. *IEEE Transactions on Robotics*, 39(6):4312–4331, 2023.

[81] Zhicheng Zhou, Cheng Zhao, Daniel Adolfsson, Songzhi Su, Yang Gao, Tom Duckett, and Li Sun. Ndt-transformer: Large-scale 3d point cloud localisation using the normal distribution transform representation. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 5654–5660. IEEE, 2021.

[82] Maximilian Hilger, Nils Mandischer, and Burkhard Corves. Randt slam: Radar slam based on intensity-augmented normal distributions transform. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7831–7838. IEEE, 2024.

[83] Erik Einhorn and Horst-Michael Gross. Generic ndt mapping in dynamic environments and its application for lifelong slam. *Robotics and Autonomous Systems*, 69: 28–39, 2015.

[84] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE, 2023.

[85] Yue Pan, Xingguang Zhong, Louis Wiesmann, Thorbjörn Posewsky, Jens Behley, and Cyrill Stachniss. Pin-slam: Lidar slam using a point-based implicit neural representation for achieving global map consistency. *IEEE Transactions on Robotics*, 2024.

[86] Yuhang Ming, Di Ma, Weichen Dai, Han Yang, Rui Fan, Guofeng Zhang, and Wanzeng Kong. Slc^2 -slam: Semantic-guided loop closure using shared latent code for nerf slam. *IEEE Robotics and Automation Letters*, 2025.

[87] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024.

[88] Lisong C Sun, Neel P Bhatt, Jonathan C Liu, Zhiwen Fan, Zhangyang Wang, Todd E Humphreys, and Ufuk Topcu. Mm3dgs slam: Multi-modal 3d gaussian splatting for slam using vision, depth, and inertial measurements. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10159–10166. IEEE, 2024.

[89] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024.

[90] Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten, and Seth Teller. An atlas framework for scalable mapping. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1899–1906. IEEE, 2003.

[91] Shoudong Huang, Zhan Wang, and Gamini Dissanayake. Sparse local submap joining filter for building large-scale maps. *IEEE Transactions on Robotics*, 24(5):1121–1130, 2008.

[92] Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Linear slam: A linear solution to the feature-based and pose graph slam based on submap joining. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 24–30. IEEE, 2013.

[93] Jun Wang, Jingwei Song, Liang Zhao, Shoudong Huang, and Rong Xiong. A submap joining algorithm for 3d reconstruction using an rgb-d camera based on point and plane features. *Robotics and Autonomous Systems*, 118:93–111, 2019.

[94] Jiaheng Zhao, Tiancheng Li, Tong Yang, Liang Zhao, and Shoudong Huang. 2d laser slam with closed shape features: Fourier series parameterization and submap joining. *IEEE Robotics and Automation Letters*, 6(2):1527–1534, 2021.

[95] René Wagner, Udo Frese, and Berthold Bäuml. Graph slam with signed distance function maps on a humanoid robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2691–2698. IEEE, 2014.

[96] Bing-Jui Ho, Paloma Sodhi, Pedro Teixeira, Ming Hsiao, Tushar Kusnur, and Michael Kaess. Virtual occupancy grid map for submap-based pose graph slam and planning in 3d environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2175–2182. IEEE, 2018.

[97] Alexander Millane, Zachary Taylor, Helen Oleynikova, Juan Nieto, Roland Siegwart, and César Cadena. C-blox: A scalable and consistent tsdf-based dense mapping approach. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 995–1002. IEEE, 2018.

[98] V A Prisacariu, O Kähler, S Golodetz, M Sapienza, T Cavallari, P H S Torr, and D W Murray. InfiniTAM v3: A Framework for Large-Scale 3D Reconstruction with Loop Closure. *arXiv pre-print arXiv:1708.00783v1*, 2017.

[99] Jakob Engel, Juan D. Tard´os, Javier Civera, Margarita Chli, Stefan Leutenegger, Frank Dellaert, and Daniel Cremers. *Visual SLAM*. Cambridge University Press.

[100] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, pages 298–372. Springer, 2000.

[101] Kurt Konolige and Motilal Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[102] Dieter Sibley, Christopher Mei, Ian D Reid, and Paul Newman. Adaptive relative bundle adjustment. In *Robotics: science and systems*, volume 32, page 33, 2009.

[103] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[104] Liang Zhao, Shoudong Huang, Yanbiao Sun, Lei Yan, and Gamini Dissanayake. Parallaxba: bundle adjustment using parallax angle feature parametrization. *The International Journal of Robotics Research*, 34(4-5):493–516, 2015.

[105] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[106] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[107] Zhehua Mao, Liang Zhao, Shoudong Huang, Yiting Fan, and Alex Pui-Wai Lee. Direct bundle adjustment for 3d image fusion with application to transesophageal echocardiography. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 548–554. IEEE, 2021.

[108] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611. IEEE, 2015.

[109] Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. Keyframe-based dense planar slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5110–5117. Ieee, 2017.

[110] Alexander JB Trevor, John G Rogers, and Henrik I Christensen. Planar surface slam with 3d and 2d sensors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3041–3048. IEEE, 2012.

[111] Patrick Geneva, Kevin Eckenhoff, Yulin Yang, and Guoquan Huang. Lips: Lidar-inertial 3d plane slam. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 123–130. IEEE, 2018.

[112] Lipu Zhou, Shengze Wang, and Michael Kaess. π-lsam: Lidar smoothing and mapping with planes. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 5751–5757. IEEE, 2021.

[113] Lipu Zhou, Daniel Koppel, and Michael Kaess. Lidar slam with plane adjustment for indoor environment. *IEEE Robotics and Automation Letters*, 6(4):7073–7080, 2021.

[114] Zheng Liu and Fu Zhang. Balm: Bundle adjustment for lidar mapping. *IEEE Robotics and Automation Letters*, 6(2):3184–3191, 2021.

[115] Zheng Liu, Xiyuan Liu, and Fu Zhang. Efficient and consistent bundle adjustment on lidar point clouds. *IEEE Transactions on Robotics*, 2023.

[116] Xiyuan Liu, Zheng Liu, Fanze Kong, and Fu Zhang. Large-scale lidar consistent mapping using hierarchical lidar bundle adjustment. *IEEE Robotics and Automation Letters*, 8(3):1523–1530, 2023.

[117] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. *Georgia Institute of Technology, Tech. Rep*, 2:4, 2012.

[118] Shuai Zhang, Liang Zhao, Shoudong Huang, Evangelos B Mazomenos, and Danail Stoyanov. Direct camera-only bundle adjustment for 3d textured colon surface reconstruction based on pre-operative model. *IEEE Transactions on Medical Robotics and Bionics*, 2024.

[119] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019.

[120] Yiran Zhou, Yingyu Wang, Shoudong Huang, and Liang Zhao. Correspondence-free multiview point cloud registration via depth-guided joint optimisation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025.

[121] Martin C. Martin and Hans P. Moravec. Robot evidence grids. Technical report, The Robotics Institute, Carnegie-Mellon University, 1996.

[122] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[123] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[124] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[125] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003. URL `http://radish.sourceforge.net/`.

[126] Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

[127] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[128] Milad Ramezani, Yiduo Wang, Marco Camurri, David Wisth, Matias Mattamala, and Maurice Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4353–4360, 2020. doi: 10.1109/IROS45743.2020.9340849.

[129] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017.

[130] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6(61):2783, 2021.

[131] Fuzhen Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.

[132] Liang Zhao, Zhehua Mao, and Shoudong Huang. Feature-Based SLAM: Why Simultaneous Localisation and Mapping? In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.009.

[133] Jiaheng Zhao, Liang Zhao, Shoudong Huang, and Yue Wang. 2d laser slam with general features represented by implicit functions. *IEEE Robotics and Automation Letters*, 5(3):4329–4336, 2020.

[134] Kai Ni, Drew Steedly, and Frank Dellaert. Tectonic sam: Exact, out-of-core, submap-based slam. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1678–1685. IEEE, 2007.

[135] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[136] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)*, 32(3):1–22, 2013.