SDF SLAM

Robin Jehn

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2025

Abstract

This dissertation presents a novel SLAM framework that leverages a continuous signed distance function (SDF) to jointly optimize sensor poses and map reconstruction within a unified optimization formulation. Traditional SLAM methods typically decouple pose estimation and mapping into separate stages, which can lead to cumulative errors and inconsistencies, particularly in feature-poor or dynamic environments. In contrast, the proposed approach simultaneously estimates both the robot trajectory and the continuous map by directly optimizing the SDF values defined over a discrete grid. This unified framework incorporates geometric constraints, such as the Eikonal condition, and introduces hallucinated points along sensor beams to mitigate the adverse effects of sensor noise and sparse data.

The optimization problem is formulated as a nonlinear least squares task and is solved efficiently using the Levenberg–Marquardt algorithm, exploiting the inherent sparsity of the Jacobian matrix to ensure computational tractability.

The experimental results show that the unified SDF framework exhibits a degree of robustness to noise and generates coherent maps while maintaining acceptable computational tractability. Notably, the choice of the finite difference scheme significantly affects the performance, with forward differencing yielding better results in terms of stability and convergence. However, in its current state, the optimization does not adequately address drift, as no explicit loop closure [11] mechanism is implemented.

While the initial outcomes are encouraging, further refinements are necessary to fully match the performance levels of state-of-the-art algorithms. Future research will focus on optimizing the computational aspects and addressing remaining challenges to achieve more accurate and reliable real-time SLAM performance across diverse environments.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Robin Jehn*)

Acknowledgements

I would like to thank Dr. Liang Zhao for the opportunity of writing this dissertation and his help.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background & Motivation

Simultaneous Localization and Mapping (SLAM) is a key technology in robotics and autonomous systems that allows machines to explore and navigate unknown environments [54]. It works by building a map of the surroundings and at the same time figuring out the location of the machine within that map. To do this, SLAM combines data from sensors such as LiDAR, cameras, and radar. As the machine moves, it continuously updates both the map and its position. SLAM is often called a ”chicken or egg” problem because if you already had a perfect map, you could figure out where you are, and if you always knew your exact location, you could stitch together sensor readings to create a map. The challenge is that both the map and the position must be estimated at the same time, making SLAM difficult to solve [17].

SLAM is used in many different fields, with one of the most well-known applications being self-driving cars [55]. It helps vehicles understand their surroundings, avoid obstacles, and navigate safely without needing pre-existing maps. But SLAM is also behind many other technologies we use daily. You may have noticed that Google Maps sometimes asks you to point your phone's camera around to improve your location accuracy - this uses SLAM to recognize landmarks and match them to known map data. In augmented and virtual reality (AR/VR), it allows devices to track their position in real-time, making it possible to place digital objects into the real world [26, 4]. Another important use is in 3D scanning and reconstruction, where SLAM helps create detailed 3D models of environments for architecture, film production, and game development [39].

Despite its many applications, SLAM still faces several challenges that limit its accuracy and reliability in real-world scenarios [30, 41, 9]. One major issue is sensor noise, where inaccuracies in LiDAR, cameras, or IMUs can lead to errors in position estimation and mapping [41]. These errors accumulate over time, causing drift, where the estimated position gradually deviates from reality [30]. Additionally, dynamic environments pose a challenge, as most SLAM algorithms assume a static world [7], making it difficult to handle moving objects like pedestrians or vehicles. Another limitation is computational efficiency, many high-accuracy SLAM methods require significant processing power,

making real-time performance difficult, especially on resource-limited devices like smartphones or drones [38]. SLAM also struggles in feature-poor environments [34], such as long corridors or open fields, where there are few distinct objects to use for localization. In some applications, such as augmented reality, poor SLAM performance may simply result in a frustrating user experience. However, in self-driving cars, inaccurate localization can cause serious accidents and even fatalities [57]. Improving SLAM is therefore not just about saving time or reducing costs, but also about our safety.

It is important to acknowledge the dual-use nature of SLAM [60]. It can be applied for the aforementioned beneficial purposes, however, the same technology can also be used in military applications, surveillance, and autonomous weapon system. For example, while SLAM enhances robotic navigation in disaster response [61], it can also improve the capabilities of autonomous drones in warfare [59].

1.2 Problem statement

Conventional SLAM methods usually employ a two-step process [54, 17]: first, they compute the most probable trajectory from sensor data, and then they build a map using the estimated poses. Although this approach is widely used, it inherently separates pose estimation and mapping into distinct optimization tasks. As a result, any errors in estimating the trajectory propagate into the final map [54]. This can lead to suboptimal performance.

To address these limitations, this research proposes a joint optimization approach where the poses and the map are simultaneously estimated within a single least-squares optimization framework, similarly to the work of Zhao et al.. Instead of treating pose estimation and mapping as separate problems, this method formulates SLAM as a unified optimization task that minimizes a single cost function incorporating both components.

A key benefit of this optimization-based approach is its flexibility in incorporating additional constraints. Since the SLAM problem is formulated as a single least-squares cost function, it becomes straightforward to add new constraints to enforce desirable properties. For example, constraints can be introduced to:

- Ensure smoothness in the map representation to prevent excessive noise in surface reconstructions.
- Incorporate loop closure corrections, allowing previously visited areas to be recognized and used to correct accumulated drift.

This flexibility makes the approach highly adaptable to different SLAM scenarios, ranging from dense indoor mapping to large-scale outdoor navigation. By integrating map and pose estimation into a single optimization process, this work aims to overcome the limitations of traditional SLAM, leading to more accurate, robust, and computationally efficient solutions.

1.3 Objectives & Research Questions

In this study, we present a proof-of-concept SLAM framework that leverages a continuous SDF to jointly optimize both the map representation and the sensor poses within a single unified optimization process. Rather than aiming for an immediate competitive performance, our primary goal is to assess whether this integrated formulation shows promise and merits further investigation. To this end, we focus on the following key questions:

1. **Is a unified optimization formulation promising?**
   By integrating map reconstruction and sensor pose estimation into a single optimization framework, we hypothesize that the interdependencies between these components can be better exploited to reduce error accumulation and yield a more coherent map.
2. **Can a continuous SDF yield accurate map reconstructions?**
   We analyze the behavior of a continuous SDF for environment representation by evaluating the quality of the reconstructed maps.
3. **Is the overall approach computationally feasible?**
   Finally, we assess the computational tractability of the proposed framework by analyzing the sparsity of the Jacobian matrix and the convergence behavior of the Levenberg–Marquardt algorithm.

By addressing these questions, our work aims to provide a comprehensive assessment of the potential and limitations of the proposed SDF-based SLAM framework, serving as an initial step toward its further development.

1.4 Contributions

This work provides a proof-of-concept for a novel SLAM framework that jointly optimizes a continuous signed distance function map and sensor poses within a unified optimization scheme. Our contributions are as follows:

1. **Continuous Map Representation:** We propose an environment representation based on a continuous SDF. Unlike traditional discrete grid-based maps, our formulation leverages a continuous and quasi differentiable map model, enabling analysis of the SDF behavior within a unified framework.
2. **Efficient Optimization Strategy:** We develop an efficient optimization procedure using the Levenberg–Marquardt algorithm. By exploiting the inherent sparsity of the Jacobian matrix derived from the continuous SDF formulation, our method achieves computational tractability, demonstrating feasibility for near real-time applications.
3. **Proof-of-Concept Evaluation:** We validate the proposed framework on both simulated and real datasets. Although our experimental evaluation is preliminary, it provides insight into the behavior of the continuous SDF and the overall computational feasibility of the approach, thereby motivating further investigation.

Together, these contributions demonstrate the potential of a unified, continuous optimization approach for SLAM and lay the foundation for future work aimed at refining and extending this method.

1.5 Chapter summary

This dissertation is structured to build a comprehensive understanding of the proposed continuous SDF-based SLAM framework. A brief summary of each chapter is provided below:

- **Introduction:** This chapter introduces the concept of SLAM and its applications in robotics and autonomous systems. It outlines the primary research questions and summarizes the main contributions of this work.
- **Literature Review:** This chapter reviews the current state of the art in SLAM and related methodologies. It examines previous approaches and relevant literature to identify their strengths and limitations, thereby establishing a foundation for the proposed method.
- **Methodology:** In this chapter, the proposed SLAM framework is described in detail. It defines the objective function and details how each residual is calculated. The chapter also explains the derivation of the Jacobian for the objective function and discusses strategies for exploiting its sparsity to efficiently solve the optimisation problem.
- **Experiments & Evaluation:** This chapter presents a comprehensive experimental evaluation of the proposed continuous SDF-based SLAM framework. Detailed quantitative and qualitative analyses are provided to validate the framework's effectiveness in various scenarios.
- **Discussion & Analysis:** In this chapter, the experimental results are critically examined. It offers an in-depth discussion on the strengths and limitations of the proposed method, including the impact of key design choices and parameter settings on performance. The analysis covers issues such as convergence behavior, sensitivity to initial conditions, and computational trade-offs. The chapter also contextualizes the findings within the broader SLAM literature, highlighting how the continuous SDF representation and joint optimization approach address common challenges in SLAM and where further improvements may be required.
- **Conclusion & Future Work:** The final chapter summarizes the main contributions of the dissertation, reiterating the benefits of a unified SLAM formulation that integrates pose estimation and map reconstruction through a continuous SDF framework. It reflects on the overall impact of the work, its significance for real-world applications, and the improvements in accuracy and robustness demonstrated through the experiments. The chapter concludes by outlining potential directions for future research, such as extending the approach to three dimensions, incorporating additional sensor modalities, optimizing for real-time performance, and exploring alternative continuous map representations.

Chapter 2

Literature Review

2.1 Traditional Feature Based SLAM

Traditional feature-based SLAM [19, 5] methods rely on detecting and describing salient features [50, 56, 52], such as corners or edges, from raw sensor data using algorithms like SIFT [31], ORB [45], or SURF [6]. These features are then matched across frames to estimate the robot's motion and incrementally build a map of the environment [12]. The accuracy of these methods is highly dependent on robust feature extraction and matching. When feature matching fails, the quality of pose estimation and the resulting map deteriorates.

One of the earliest and most widely used approaches to SLAM is EKF-SLAM [18] (Extended Kalman Filter SLAM). This method models the environment as a collection of landmarks with Gaussian noise and updates their positions incrementally using the Extended Kalman Filter. The main advantage of EKF-SLAM is its ability to estimate both robot pose and map features in a probabilistic framework, making it well-suited for real-world environments with measurement uncertainty. However, its main drawback lies in computational complexity, as the algorithm requires maintaining and updating a covariance matrix that scales quadratically with the number of landmarks [18]. This limits its applicability to large-scale mapping scenarios. Some of the other problems include linearization inaccuracies and sensitivity to initial conditions [50, 12].

Particle-based approaches represent the posterior distribution over the robot's state and the map using a set of weighted particles. A prime example is FastSLAM [53], which decomposes the SLAM problem into two subproblems: one that estimates the robot's trajectory and another that independently estimates the positions of landmarks conditioned on that trajectory. Each particle carries its own hypothesis of the robot's path along with a set of landmark estimates, typically maintained as Gaussian distributions. This formulation allows FastSLAM to manage non-linearities and multi-modal distributions effectively, although it can face challenges like sample impoverishment [22] and scalability as the number of landmarks grows [40, 12].

Graph-based SLAM methods, in contrast, formulate SLAM as a nonlinear least squares optimization problem over a factor graph [16]. In this framework, nodes represent

robot poses and landmarks, while edges encode constraints derived from feature correspondences [15]. A classic example is Graph-SLAM [51], which optimizes the entire trajectory and map simultaneously by minimizing the error between predicted and observed measurements. Modern optimization frameworks such as g2o [29] and GTSAM [14] have built upon this concept to handle large-scale environments efficiently by exploiting the sparsity of the problem. Systems like ORB-SLAM [35] extend these ideas further by integrating robust feature matching with incremental pose graph optimization.

2.2 Non-Feature Based SLAM

Hector SLAM [47] is a 2D LiDAR SLAM system that directly operates on raw scan data without relying on extracted features. It employs fast scan matching by using an adaptive grid search combined with Gauss–Newton alignment [48] to match laser scans to an occupancy grid map. Operating at high update rates (around 40 Hz), Hector SLAM computes 2D pose estimates solely from LiDAR measurements in real time [42].

Google Cartographer [25] is a real-time SLAM library for both 2D and 3D mapping that also bypasses the need for explicit feature extraction. It builds small local submaps by inserting consecutive laser scans and computes scan-to-submap matches using correlative scan matching. Additionally, Cartographer fuses sensor data such as IMU and odometry when available, and continuously performs loop closure [11] and global optimization to produce globally consistent occupancy grid maps [58, 42].

Occupancy-SLAM [62] shifts from sequential map estimation to simultaneous optimization of both the robot trajectory and the environment representation. Traditional SLAM approaches often estimate the robot's trajectory first and then construct the map based on the optimized poses [25]. However, this recent method, introduced an optimization-based approach that jointly estimates both the robot's trajectory and the occupancy grid map [20]. This approach formulates the SLAM problem as a batch optimization task, where both the robot poses and occupancy values at discrete grid locations are optimized together, similar to the work of VoxGraph [43] and Kimera-PGMO [44].

This dissertation builds on the same principle of joint optimization, but instead of representing the environment as an occupancy grid, we use a SDF [36, 13]. SDFs store the signed distance to the nearest surface rather than the probability of occupancy, providing a continuous and quasi differentiable representation of the environment. Unlike occupancy grids, which rely on discrete cell-based updates, SDFs allow for interpolation across a grid, improving surface smoothness and enabling more precise geometric constraints [9, 38]. The optimization framework remains similar, both robot poses and SDF values at selected grid nodes are estimated simultaneously, but the map formulation changes fundamentally from occupancy probabilities to distance-based surface modeling.

2.3 Signed Distance Function in SLAM

The use of SDFs in SLAM has evolved considerably over the years. One of the earliest uses of the SDF concept in the context of SLAM is found in [13], where the idea of volumetric integration of range images using a truncated signed distance function (TSDF) was introduced. This work laid the foundation for many subsequent dense mapping techniques by providing a continuous representation of the environment that naturally encodes surface geometry.

Building on this idea, [36] introduced KinectFusion, which was the first real-time dense SLAM system based on TSDFs. KinectFusion demonstrated that by fusing depth measurements into a TSDF volume and performing iterative alignment, it was possible to achieve both dense mapping and accurate tracking in real time. The success of KinectFusion spurred a number of extensions and improvements aimed at increasing robustness and scaling to larger environments.

In parallel to these volumetric approaches, researchers began exploring the use of SDFs for scan registration in 2D scenarios. The work in [21] presents a SLAM frontend that uses the signed distance function to register laser scans. Unlike traditional methods that rely on explicit correspondence association or joint optimization of scan points, this approach directly leverages the SDF to align scans.

VoxGraph [43] introduces a 3D SLAM system that partitions the environment into multiple TSDF submaps rather than relying on a single monolithic volume. Each submap is a localized TSDF with its own origin, storing distance and weight information for each voxel. As the robot moves, new submaps are created to limit drift, and when they overlap, constraints are computed by comparing their SDF content, via euclidean SDF generation and surface point extraction, so that a pose graph optimization can align them globally, effectively performing loop closure at the submap level.

More recently, a multi-robot SDF-SLAM system [27] has shown that an SDF-based map representation can significantly reduce drift. In this system, the continuous nature of the SDF not only allows for smoother surface reconstructions but also tends to yield minimal drift over long trajectories. In many cases, the high fidelity of the SDF map alleviates the need for extensive loop closure detection, as the map inherently maintains global consistency.

Another notable recent development is the use of neural networks in SLAM [49, 37, 38, 10]. PIN-SLAM [38] uses a point-based implicit neural representation to achieve globally consistent mapping while maintaining a compact memory footprint. Rather than storing explicit grid-based SDF values, PIN-SLAM represents the environment as a sparse set of neural points with associated latent features that are optimized online [3]. Similar to our work, PIN-SLAM enforces geometric consistency through the Eikonal equation [23, 8], ensuring that the gradient of the SDF has a unit norm. However, while PIN-SLAM relies on a complex neural network pre-training and feature optimization scheme, our approach builds on the same fundamental idea—geometric consistency enforced via the Eikonal constraint, but achieves it within a simpler and more direct optimization framework.

Chapter 3

Methodology

In addressing the SLAM problem, we propose an approach that jointly estimates a continuous map representation, via a SDF, and the sensor transformations. The formulation enforces geometric consistency by incorporating the Eikonal [8] condition. The approach results in a sparse optimization problem that can be solved efficiently.

3.1 Problem Definition and Parameters

We consider a continuous signed distance function

$$D : \Omega \to \mathbb{R}, \tag{3.1}$$

defined on a bounded domain $\Omega \subset \mathbb{R}^2$ that represents the area of interest. For any point $\mathbf{x} \in \Omega$, the SDF is given by

$$D(\mathbf{x}) = \begin{cases} +d(\mathbf{x}, S) & \text{if } \mathbf{x} \text{ is in front of the surface,} \\ -d(\mathbf{x}, S) & \text{if } \mathbf{x} \text{ is behind the surface,} \end{cases} \tag{3.2}$$

where $d(\mathbf{x}, S)$ denotes the Euclidean distance from $\mathbf{x}$ to the closest surface S.

In practice, $D(\mathbf{x})$ is approximated on a discrete grid. Let the grid be defined by

$$m_{w,h} = (w, h), \quad w = 0, 1, \ldots, l_w - 1, \quad h = 0, 1, \ldots, l_h - 1, \tag{3.3}$$

so that the grid covers the domain Ω and contains a total of

$$n_{\text{grid}} = l_w \times l_h \tag{3.4}$$

nodes. The discrete SDF, denoted by $\hat{D}$, is defined at each node as

$$\hat{D}(m_{w,h}) \approx D(\mathbf{x}_{w,h}), \tag{3.5}$$

where $\mathbf{x}_{w,h}$ are the coordinates corresponding to node $m_{w,h}$.

To approximate D at any arbitrary point $\mathbf{x} = (x, y) \in \Omega$, we use bilinear interpolation of the grid values. Assume that $\mathbf{x}$ lies in the cell defined by the nodes $m_{w,h}$, $m_{w+1,h}$, $m_{w,h+1}$, and $m_{w+1,h+1}$. If the point lies on a grid line we take the cell to the right and/or top. Define

$$\alpha = \frac{x - x_w}{x_{w+1} - x_w}, \quad \beta = \frac{y - y_h}{y_{h+1} - y_h}. \tag{3.6}$$

Then, the interpolated SDF value is given by

$$\begin{aligned} D(x,y) = (1-\alpha)(1-\beta)\hat{D}(m_{w,h}) + \alpha(1-\beta)\hat{D}(m_{w+1,h}) \\ + (1-\alpha)\beta\hat{D}(m_{w,h+1}) + \alpha\beta\hat{D}(m_{w+1,h+1}). \end{aligned} \tag{3.7}$$

Figure 3.1 illustrates the bilinear interpolation process.

In addition to the map representation, our SLAM problem requires estimating the sensor (or robot) transformations. For a sequence of N LiDAR scans, let the pose corresponding to scan i be expressed as

$$X_r^i = \begin{bmatrix} \mathbf{t}_i \\ \theta_i \end{bmatrix} \in \mathbb{R}^3, \tag{3.8}$$

where $\mathbf{t}_i \in \mathbb{R}^2$ is the translation vector and $\theta_i \in \mathbb{R}$ is the rotation angle. The first pose X_r^0 is fixed (e.g., $X_r^0 = [0,0,0]^T$), and the remaining $N-1$ poses are estimated.

Thus, the overall state vector $\mathbf{X}$ is defined as

$$\mathbf{X} \triangleq \left[\hat{D}(m_{0,0}), \hat{D}(m_{0,1}), \ldots, \hat{D}(m_{l_w-1,l_h-1}), X_r^1, X_r^2, \ldots, X_r^{N-1}, \right]^T, \tag{3.9}$$

with a total dimension of

$$\dim(\mathbf{X}) = l_w l_h + 3 \times (N-1). \tag{3.10}$$

This state vector comprises both the grid values $\{\hat{D}(m_{w,h})\}$ and the sensor transformations $\{X_r^i\}_{i=1}^{N-1}$, forming the basis for our joint optimization framework.

3.2 Optimization Formulation

We formulate the SLAM problem as a nonlinear least-squares (NLLS) optimization to estimate the state vector $\mathbf{X}$ that best explains the observed LiDAR data and odometry measurements. Recall that the state $\mathbf{X}$ comprises both the grid values $\{\hat{D}(m_{w,h})\}$ and the robot poses $\{X_r^i\}_{i=1}^{N-1}$ (with X_r^0 fixed). Our overall objective function is defined as a weighted sum of four residual terms:

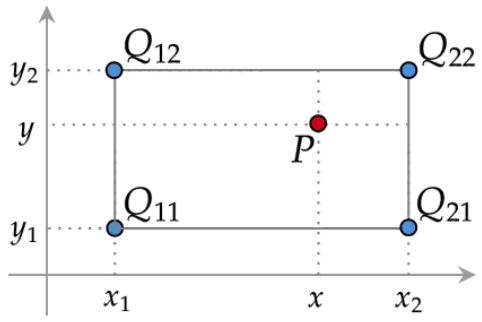


Figure 3.1: Illustration of bilinear interpolation [1] for the continuous signed distance function. The query point $P = (x, y)$ lies within the grid cell defined by the four nodes $Q_{11} = \mathbf{x}_{w,h}$, $Q_{21} = \mathbf{x}_{w+1,h}$, $Q_{12} = \mathbf{x}_{w,h+1}$, and $Q_{22} = \mathbf{x}_{w+1,h+1}$. The interpolated value at P is computed as a weighted sum of the SDF values at these nodes, with weights determined by the relative distances along the x and y directions.

$$F(\mathbf{X}) = w_s F_s(\mathbf{X}) + w_h F_h(\mathbf{X}) + w_e F_e(\mathbf{X}) + w_o F_o(\mathbf{X}), \tag{3.11}$$

where w_s, w_h, w_e, and w_o denote the weights for the scan, hallucination, Eikonal, and odometry residuals, respectively.

3.2.1 Scan Residual

Scan measurements correspond to the first contact with an object, so the SDF value at a transformed scan point should be zero. Let $\mathbf{p}_s$ be a scan point in the sensor frame from scan i, and let T_i be its associated rigid-body transformation. The global coordinates are obtained via

$$\mathbf{p}_g = T_i \mathbf{p}_s. \tag{3.12}$$

Since a scan point $\mathbf{p}_s$ represents a surface contact, the expected SDF value at $\mathbf{p}_g$ is zero. Hence, the residual for each scan point is defined as

$$r_s(\mathbf{p}_s) = D(T_i \mathbf{p}_s). \tag{3.13}$$

The scan residual term is then the sum of squared residuals over all scan points:

$$F_s(\mathbf{X}) = \sum_{k \in I_{\text{scan}}} \sum_{\mathbf{p}_s \in S_k} \Big(r_s(\mathbf{p}_s) \Big)^2. \tag{3.14}$$

3.2.2 Hallucination Residual

To prevent degenerate solutions (e.g., setting $\hat{D}(m_{w,h}) = 0$ everywhere) that may occur when relying solely on scan points, we introduce *hallucinated points* along each sensor

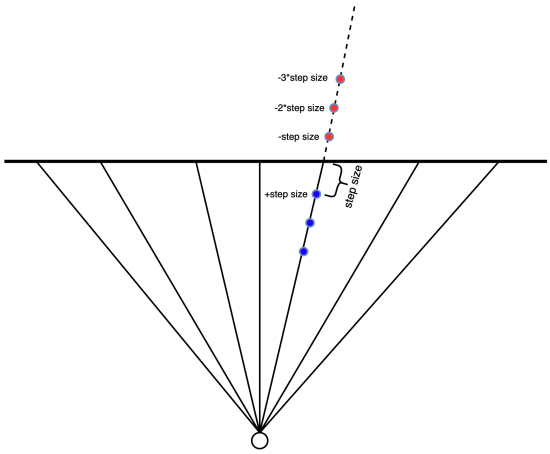


Figure 3.2: Illustration of hallucinated points along a sensor beam. Blue and red markers indicate hallucinated points sampled at regular step intervals from the measured surface. Each point is assigned an expected SDF value $\delta(x)$, which is positive when the point lies outside the object and negative when it lies inside.

beam. These points are generated along each ray at a distance x from the measured scan point. Unlike scan points, which are expected to lie exactly on the surface (i.e., $D = 0$), hallucinated points are assigned an expected SDF value $\delta(x)$ that approximates the true signed distance at that location.

We define the expected value $\delta(x)$ as

$$\delta(x) = \begin{cases} x, & \text{if the point is outside the object,} \\ -x, & \text{if the point is inside the object.} \end{cases} \tag{3.15}$$

Similarly to the scan residual, the residual for a hallucinated point is computed by transforming the point from the sensor frame to the global frame and then evaluating the SDF via bilinear interpolation. Thus, for a hallucinated point $\mathbf{p}_s^{\text{hall}}$, the residual is defined as

$$r_h(\mathbf{p}_s^{\text{hall}}) = D(T_i \mathbf{p}_s^{\text{hall}}) - \delta(x). \tag{3.16}$$

The overall hallucination residual term in the objective function is given by

$$F_h(\mathbf{X}) = \sum_{k \in I_{\text{halluc}}} \sum_{\mathbf{p}_s^{\text{hall}} \in H_k} \left(r_h(\mathbf{p}_s^{\text{hall}}) \right)^2. \tag{3.17}$$

3.2.3 Eikonal Residual

The SDF is required to satisfy the Eikonal condition, ensuring that the gradient magnitude is unity in the direction of the closest surface. At each grid node $m_{w,h}$, we approximate the gradients using finite differences:

$$\left(\frac{\partial \hat{D}}{\partial x}\right)_{w,h} \approx \frac{\hat{D}(m_{w+1,h}) - \hat{D}(m_{w,h})}{\Delta x}, \quad \left(\frac{\partial \hat{D}}{\partial y}\right)_{w,h} \approx \frac{\hat{D}(m_{w,h+1}) - \hat{D}(m_{w,h})}{\Delta y}. \tag{3.18}$$

Let $\mathbf{n}_{w,h} = (n_x(m_{w,h}), n_y(m_{w,h}))$ denote the estimated unit vector pointing towards the nearest surface at grid node $m_{w,h}$. To compute $n_{w,h}$, we first estimate the normal at each scan point using the PCL library [46], which determines the k closest neighbors and applies Principal Component Analysis (PCA) [33] to compute the normal. The normal is defined as the second principal component oriented towards the origin. Finally, the surface normal $n_{w,h}$ is assigned as the normal of the scan point that is closest to node $m_{w,h}$. The Eikonal residual at $m_{w,h}$ is then defined as

$$r_e(m_{w,h}) = 1 - \mathbf{n}_{w,h} \cdot \nabla \hat{D}(m_{w,h}). \tag{3.19}$$

The overall Eikonal residual term is

$$F_e(\mathbf{X}) = \sum_{w=0}^{l_w-1} \sum_{h=0}^{l_h-1} \Big(r_e(m_{w,h})\Big)^2. \tag{3.20}$$

3.2.4 Odometry Residual

Relative odometry measurements provide constraints between consecutive frames. Let the measured odometry between frame i and frame $i+1$ be denoted by

$$\mathbf{O}_i^{\text{meas}} = \begin{bmatrix} O_{i,x}^{\text{meas}} \\ O_{i,y}^{\text{meas}} \\ O_{i,\theta}^{\text{meas}} \end{bmatrix} \in \mathbb{R}^3, \tag{3.21}$$

where $O_{i,x}^{\text{meas}}$ and $O_{i,y}^{\text{meas}}$ represent the measured translation components and $O_{i,\theta}^{\text{meas}}$ represents the measured rotation.

Given the estimated poses T_i and T_{i+1}, where each pose is defined by a rotation $R(\theta_i)$ and a translation $\mathbf{t}_i$, we compute the estimated relative translation by transforming the difference $\mathbf{t}_{i+1} - \mathbf{t}_i$ into the coordinate frame of pose i:

$$\Delta \mathbf{t}_i^{\text{est}} = R(-\theta_i)(\mathbf{t}_{i+1} - \mathbf{t}_i) = \begin{bmatrix} \Delta x_i^{\text{est}} \\ \Delta y_i^{\text{est}} \end{bmatrix}. \tag{3.22}$$

The estimated relative rotation is given by

$$\Delta \theta_i^{\text{est}} = \theta_{i+1} - \theta_i. \tag{3.23}$$

Thus, the estimated relative transformation is expressed as

$$\hat{\mathbf{O}}_i = \begin{bmatrix} \Delta x_i^{\text{est}} \\ \Delta y_i^{\text{est}} \\ \Delta \theta_i^{\text{est}} \end{bmatrix}. \tag{3.24}$$

We then define three independent odometry residuals, one for each parameter:

$$r_{o,x}^i = \Delta x_i^{\text{est}} - O_{i,x}^{\text{meas}}, \quad r_{o,y}^i = \Delta y_i^{\text{est}} - O_{i,y}^{\text{meas}}, \quad r_{o,\theta}^i = \Delta \theta_i^{\text{est}} - O_{i,\theta}^{\text{meas}}. \tag{3.25}$$

Finally, the overall odometry residual term in the objective function is formulated as the sum of the squared residuals for each parameter:

$$F_o(\mathbf{X}) = \sum_{i=1}^{N-1} \left[(r_{o,x}^i)^2 + (r_{o,y}^i)^2 + (r_{o,\theta}^i)^2 \right], \tag{3.26}$$

which penalizes discrepancies between the estimated relative motion and the corresponding measured odometry on a per-parameter basis.

3.3 Optimization Technique

We solve the nonlinear least-squares problem

$$\min_{\mathbf{X}} F(\mathbf{X}) = w_s F_s(\mathbf{X}) + w_h F_h(\mathbf{X}) + w_e F_e(\mathbf{X}) + w_o F_o(\mathbf{X}) \tag{3.27}$$

using the Levenberg–Marquardt (LM) algorithm [32].

At each iteration k, the residual functions are linearized about the current estimate $\mathbf{X}^{(k)}$ to yield the normal equations

$$\left(J^T J + \lambda I\right) \Delta \mathbf{X} = -J^T \tilde{\mathbf{r}}, \tag{3.28}$$

where

- J is the Jacobian matrix of the weighted residuals (with $\tilde{r}_i(\mathbf{X}) = \sqrt{w_i}\, r_i(\mathbf{X})$) evaluated at $\mathbf{X}^{(k)}$,
- $\tilde{\mathbf{r}}$ is the vector of weighted residuals,
- λ is a damping parameter that interpolates between the Gauss–Newton method (small λ) and gradient descent (large λ), and
- $\Delta \mathbf{X}$ is the computed update for the state.

The state is then updated as

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \Delta \mathbf{X}. \tag{3.29}$$

The damping parameter λ is adjusted adaptively to ensure robust convergence. This iterative LM optimization simultaneously refines both the grid SDF values and the robot poses, yielding a solution that satisfies the scan, hallucination, Eikonal, and odometry constraints.

Note that in the overall optimization the square of the residuals is scaled by the weights, so the partial derivatives have to be multiplied by the square root of the respective weight. This is omitted for simplicity in the following chapters.

3.3.1 Jacobian Computation for Scan Points Residuals

Remember that for each scan point $\mathbf{p}_s$ from scan frame i, the unweighted residual is defined as

$$r_s(\mathbf{p}_s) = D(T_i \mathbf{p}_s). \tag{3.30}$$

Let the transformation T_i map $\mathbf{p}_s$ to $\mathbf{p}_g = (x, y)$ that lies in the grid cell defined by nodes $m_{w,h}$, $m_{w+1,h}$, $m_{w,h+1}$, and $m_{w+1,h+1}$. With interpolation weights

$$\alpha = \frac{x - x_w}{x_{w+1} - x_w}, \quad \beta = \frac{y - y_h}{y_{h+1} - y_h}, \tag{3.31}$$

the derivatives of $D(x, y)$ with respect to the four grid values are given by the corresponding interpolation weights:

$$\frac{\partial D}{\partial \hat{D}(m_{w,h})} = (1-\alpha)(1-\beta), \quad \frac{\partial D}{\partial \hat{D}(m_{w+1,h})} = \alpha(1-\beta), \tag{3.32}$$

$$\frac{\partial D}{\partial \hat{D}(m_{w,h+1})} = (1-\alpha)\beta, \quad \frac{\partial D}{\partial \hat{D}(m_{w+1,h+1})} = \alpha\beta. \tag{3.33}$$

Similarly, the partial derivatives of the predicted point coordinates with respect to the transformation parameters are:

$$x = \cos\phi\, p_x - \sin\phi\, p_y + t_x, \quad y = \sin\phi\, p_x + \cos\phi\, p_y + t_y, \tag{3.34}$$

with

$$\frac{\partial x}{\partial \phi} = -\sin\phi\, p_x - \cos\phi\, p_y, \quad \frac{\partial y}{\partial \phi} = \cos\phi\, p_x - \sin\phi\, p_y, \tag{3.35}$$

$$\frac{\partial x}{\partial t_x} = 1, \quad \frac{\partial y}{\partial t_x} = 0, \quad \frac{\partial x}{\partial t_y} = 0, \quad \frac{\partial y}{\partial t_y} = 1. \tag{3.36}$$

The spatial derivatives $\partial D/\partial x$ and $\partial D/\partial y$ are computed as in Section 3.2.3. By the chain rule, the derivatives with respect to the transformation parameters are given by

$$\frac{\partial r_s}{\partial \phi} = \frac{\partial D}{\partial x}\frac{\partial x}{\partial \phi} + \frac{\partial D}{\partial y}\frac{\partial y}{\partial \phi}, \tag{3.37}$$

$$\frac{\partial r_s}{\partial t_x} = \frac{\partial D}{\partial x}, \quad \frac{\partial r_s}{\partial t_y} = \frac{\partial D}{\partial y}. \tag{3.38}$$

These contributions appear only for the four grid nodes defining the cell containing $T_i(\mathbf{p}_s)$ and the three transformation parameters associated with scan frame i.

For all other parameters in the state vector, the partial derivative is zero.

3.3.2 Jacobian Computation for Hallucinated Points Residuals

The computation for hallucinated points is analogous to that for scan points residuals. The only difference is that the unweighted residual is $r_h(\mathbf{p}_s^{hall}) = D(T_i\mathbf{p}_s^{hall}) - \delta(x)$ (with $\delta(x)$ being constant with respect to the state). Hence, the non-zero Jacobian entries are obtained exactly as in the scan points case, i.e., contributions from the four grid nodes and the three transformation parameters, with the derivative of $\delta(x)$ being zero.

3.3.3 Jacobian Computation for Eikonal Residuals

Recall that at each grid node $m_{w,h}$, the Eikonal residual is defined as

$$r_e(m_{w,h}) = 1 - \mathbf{n}_{w,h} \cdot \nabla\hat{D}(m_{w,h}), \tag{3.39}$$

The gradient $\nabla\hat{D}(m_{w,h})$ is approximated using forward differences as in Section 3.2.3. Differentiating $r_e(m_{w,h})$ with respect to the grid values yields

$$\frac{\partial r_e(m_{w,h})}{\partial m_{w,h}} = \frac{n_x m_{w,h}}{\Delta x} + \frac{n_y m_{w,h}}{\Delta y}, \tag{3.40}$$

$$\frac{\partial r_e(m_{w,h})}{\partial m_{w+1,h}} = -\frac{n_x m_{w,h}}{\Delta x}, \tag{3.41}$$

$$\frac{\partial r_e(m_{w,h})}{\partial m_{w,h+1}} = -\frac{n_y m_{w,h}}{\Delta y}. \tag{3.42}$$

For all other grid nodes, the derivative is zero. Note that this residual depends solely on the grid parameters and does not involve any transformation parameters.

3.3.4 Jacobian Computation for Odometry Residuals

We express the estimated relative transformation between frames i and $i+1$ as

$$\hat{\mathbf{O}}_i = \begin{bmatrix} \Delta \mathbf{t}_i^{\text{est}} \\ \Delta \theta_i^{\text{est}} \end{bmatrix} = \begin{bmatrix} R(-\theta_i)(\mathbf{t}_{i+1} - \mathbf{t}_i) \\ \theta_{i+1} - \theta_i \end{bmatrix}, \tag{3.43}$$

where the rotation matrix $R(-\theta_i)$ is defined as

$$R(-\theta_i) = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix}. \tag{3.44}$$

We now compute the Jacobian of $\hat{\mathbf{O}}_i$ with respect to the pose parameters in $X_i = \{\mathbf{t}_i, \theta_i\}$ and $X_{i+1} = \{\mathbf{t}_{i+1}, \theta_{i+1}\}$.

For the translation component,

$$\Delta \mathbf{t}_i^{\text{est}} = R(-\theta_i)(\mathbf{t}_{i+1} - \mathbf{t}_i), \tag{3.45}$$

differentiation with respect to $\mathbf{t}_i$ and $\mathbf{t}_{i+1}$ yields

$$\frac{\partial \Delta \mathbf{t}_i^{\text{est}}}{\partial \mathbf{t}_i} = -R(-\theta_i), \quad \frac{\partial \Delta \mathbf{t}_i^{\text{est}}}{\partial \mathbf{t}_{i+1}} = R(-\theta_i). \tag{3.46}$$

To compute the derivative of $\Delta \mathbf{t}_i^{\text{est}}$ with respect to θ_i, we differentiate the rotation matrix. Defining the skew-symmetric matrix

$$\hat{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{3.47}$$

we have

$$\frac{\partial R(-\theta_i)}{\partial \theta_i} = -R(-\theta_i)\hat{J}. \tag{3.48}$$

Thus,

$$\frac{\partial \Delta \mathbf{t}_i^{\text{est}}}{\partial \theta_i} = -R(-\theta_i)\hat{J}(\mathbf{t}_{i+1} - \mathbf{t}_i). \tag{3.49}$$

For the rotation component,

$$\Delta \theta_i^{\text{est}} = \theta_{i+1} - \theta_i, \tag{3.50}$$

the derivatives are straightforward:

$$\frac{\partial \Delta \theta_i^{\text{est}}}{\partial \theta_i} = -1, \quad \frac{\partial \Delta \theta_i^{\text{est}}}{\partial \theta_{i+1}} = 1. \tag{3.51}$$

Combining these results, the Jacobian of $\hat{\mathbf{O}}_i$ with respect to the pose $X_i = \{\mathbf{t}_i, \theta_i\}$ is given by

$$J_i = \begin{bmatrix} -R(-\theta_i) & -R(-\theta_i)\hat{J}(\mathbf{t}_{i+1} - \mathbf{t}_i) \\ \mathbf{0}_{1\times 2} & -1 \end{bmatrix}, \tag{3.52}$$

and the Jacobian with respect to the pose $X_{i+1} = \{\mathbf{t}_{i+1}, \theta_{i+1}\}$ is

$$J_{i+1} = \begin{bmatrix} R(-\theta_i) & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix}. \tag{3.53}$$

These Jacobian blocks constitute the derivative of the odometry residual

$$r_o^i = \hat{\mathbf{O}}_i - \mathbf{O}_i^{\text{meas}}, \tag{3.54}$$

with respect to the involved poses. For all other parameters in the state vector, the partial derivatives are zero.

3.3.5 Overall Jacobian Sparsity and Sparse Optimization

The full Jacobian J is assembled by stacking the individual Jacobian rows from the scan, hallucination, Eikonal, and odometry residuals. In summary:

- Each **scan point** residual contributes a row with nonzero entries for 4 grid parameters and 3 transformation parameters.
- Each **hallucinated point** residual contributes a row with nonzero entries for 4 grid parameters and 3 transformation parameters.
- Each **Eikonal** residual contributes a row with nonzero entries for only 3 grid parameters.
- Each set of three **odometry** residual contributes to three rows with a total of 12 nonzero entries.

Assuming that the state vector contains $n_x \times n_y$ grid parameters and $3 \times (N-1)$ transformation parameters, let

$$n_{\text{scan}}, \quad n_{\text{hall}}, \quad n_{\text{eik}} = (n_x - 1)(n_y - 1), \quad n_{\text{odom}} = 3(N-1) \tag{3.55}$$

denote the number of scan, hallucination, Eikonal, and odometry residuals respectively. Then, the total number of rows in J is

$$n_{\text{rows}} = n_{\text{scan}} + n_{\text{hall}} + (n_x - 1)(n_y - 1) + 3(N - 1), \tag{3.56}$$

and the number of columns is

$$n_{\text{cols}} = n_x n_y + 3(N - 1). \tag{3.57}$$

Each scan point and hallucination residual contributes 7 nonzero entries, each Eikonal residual contributes 3 nonzero entries, and each set of odometry residuals contributes 12 nonzero entries. Thus, the total number of nonzero entries is at most

$$N_{nz} = 7\Big(n_{\text{scan}} + n_{\text{hall}}\Big) + 3\,(n_x - 1)(n_y - 1) + 12\,(N - 1). \tag{3.58}$$

The total number of entries in the Jacobian is

$$N_{\text{total}} = \Big(n_{\text{scan}} + n_{\text{hall}} + (n_x - 1)(n_y - 1) + 3(N - 1)\Big) \times \Big(n_x n_y + 3(N - 1)\Big). \tag{3.59}$$

Hence, the sparsity ratio is given by

$$\text{Sparsity Ratio} = \frac{N_{nz}}{N_{\text{total}}}. \tag{3.60}$$

For example, suppose we have:

- 76 scans with a total of 45 040 scan points,
- For each scan point, 6 hallucinated points are added (so that the total number of scan-related residuals is $7 \times 45\,040$),
- A map of size 100×100 (resulting in $100 \times 100 = 10\,000$ grid parameters),
- The Eikonal residuals are computed at all grid nodes except the last row and column, i.e. 99×99 residuals,
- There are $76 - 1 = 75$ odometry residuals.

Thus, the total number of nonzero entries is:

$$N_{nz} = 7\,(7 \times 45\,040), +3\,(99 \times 99) + 12\,(75) = 2\,237\,263. \tag{3.61}$$

The number of columns is:

$$n_{\text{cols}} = 10\,000 + 225 = 10\,225. \tag{3.62}$$

The number of rows is:

$$n_{\text{rows}} = 7 \times 45\,040 + 99 \times 99 + 75 = 315\,280 + 9\,801 + 75 = 325\,156. \tag{3.63}$$

Thus, the total number of entries in J is:

$$N_{\text{total}} = n_{\text{rows}} \times n_{\text{cols}} = 325\,156 \times 10\,225 \approx 3.32472 \times 10^9. \tag{3.64}$$

Finally, the sparsity ratio is given by:

$$\text{Sparsity Ratio} = \frac{N_{nz}}{N_{\text{total}}} \approx \frac{2\,237\,263}{3.32472 \times 10^9} \approx 6.72 \times 10^{-4}. \tag{3.65}$$

This high sparsity enables efficient solution of the linear system

$$\left(J^T J + \lambda I\right) \Delta\theta = -J^T \hat{\mathbf{r}}, \tag{3.66}$$

using sparse Cholesky decomposition [28], thereby significantly reducing computational cost and memory usage during each iteration of the LM algorithm.

3.4 Incremental Optimization Algorithm

In our incremental optimization approach, new sensor frames are sequentially incorporated into the state estimate using the corresponding odometry measurements. For each new frame, the pose is computed by composing the last optimized frame with the odometry measurement, and the new frame is then appended to the state. The entire state (i.e., all frames added so far) is subsequently refined through several iterations of the Levenberg–Marquardt algorithm. In each iteration, the residuals (derived from scan, hallucination, Eikonal, and odometry terms) are computed, the Jacobian matrix is evaluated, and the normal equations are formed and solved to update the state estimate.

Algorithm 1 Batch Optimization for SLAM

Input:
- State $\mathbf{X}$ (comprising grid values $\{\hat{D}(m_{w,h})\}$, poses $\{T_i\}$)
- Maximum iterations max_iters, tolerance tol
- Initial damping λ, damping adjustment factor γ

Output: Optimized state $\mathbf{X}$

Compute initial error:

$$E_{prev} = \|\mathbf{r}(\mathbf{X})\|^2$$

for $iter = 1$ to max_iters **do**

  Compute the residual vector $\mathbf{r}(\mathbf{X})$

  Compute the Jacobian matrix J at the current state $\mathbf{X}$

  Form the normal equations:

$$\left(J^T J + \lambda I\right) \Delta\mathbf{X} = -J^T \mathbf{r}(\mathbf{X})$$

  Solve for the update $\Delta\mathbf{X}$ using sparse Cholesky factorization

  Update the state:

$$\mathbf{X} \leftarrow \mathbf{X} + \Delta\mathbf{X}$$

  Compute new error:

$$E = \|\mathbf{r}(\mathbf{X})\|^2$$

  if $\|\Delta\mathbf{X}\| < tol$ **then**

    break

  end if

  if $E < E_{prev}$ **then**

    Decrease damping:

$$\lambda \leftarrow \lambda / \gamma$$

  else

    Increase damping:

$$\lambda \leftarrow \lambda \times \gamma$$

  end if

  Set $E_{prev} \leftarrow E$

end for

Return: Optimized state $\mathbf{X}$

Algorithm 2 Incremental Optimization for SLAM Using Odometry

Input:
- Initial state $\mathbf{X}_0$ (containing grid values and an initial frame)
- Odometry measurements $\{O_1, O_2, \ldots, O_M\}$, where each O_i represents the relative transformation from the last frame to the new frame
- Increment size k (number of odometry measurements to add per iteration)
- Maximum iterations max_iters, tolerance tol
- Initial damping λ, damping adjustment factor γ

Output: Final optimized state $\mathbf{X}$ (including all frames, grid values, etc.)
Set $\mathbf{X} \leftarrow \mathbf{X}_0$
Set $i \leftarrow 1$
while $i \leq M$ **do**
    for $j = i$ to $\min(i+k-1, M)$ **do**
        $F_j = O_j \circ \text{LastFrame}(\mathbf{X})$,
        Append F_j to state $\mathbf{X}$
    end for
    Call Batch Optimization (Algorithm 1) on the updated state $\mathbf{X}$ using its current state as the initial guess.
    Update $\mathbf{X}$ with the optimized state from the batch solver.
    Set $i \leftarrow i+k$
end while
Return: Final optimized state $\mathbf{X}$

Chapter 4

Experiments & Evaluation

This chapter presents a comprehensive assessment of our SDF SLAM framework, examining its performance across various experimental scenarios to highlight both its strengths and limitations. We conduct experiments on two datasets: one simulated scan provided by Dr. Liang Zhao and one real-world dataset from the Intel Research Lab in Seattle [24]. For the simulated dataset, we assess both map accuracy and trajectory estimation quantitatively and visually. For the Intel dataset, where ground truth is unavailable, we perform a visual analysis.

Unless stated otherwise we will use the hyper-parameters defined in Table 4.1.

Objective Function	
w_s	1
w_h	1
Number of hallucinated points per scan point	6
Hallucinated point step size	0.1
w_e	1
w_o	1
Map	
Number of grid points	[100, 100]
Initial map value	0
Optimisation	
Initial Lambda	1
Lambda factor	1
Number of iterations	100

Table 4.1: Hyper-parameters used for the proposed SLAM framework

4.1 Simulated Scan Evaluation

In this section, we focus on evaluating our SLAM system using a simulated scan dataset with ground truth odometry that can be seen in Figure 4.1, enabling a precise analysis of pose accuracy and map reconstruction quality. We optimize all frames at once using the Batch Algorithm 1. We further introduce controlled noise into the odometry in separate experiments to assess the approach's robustness.

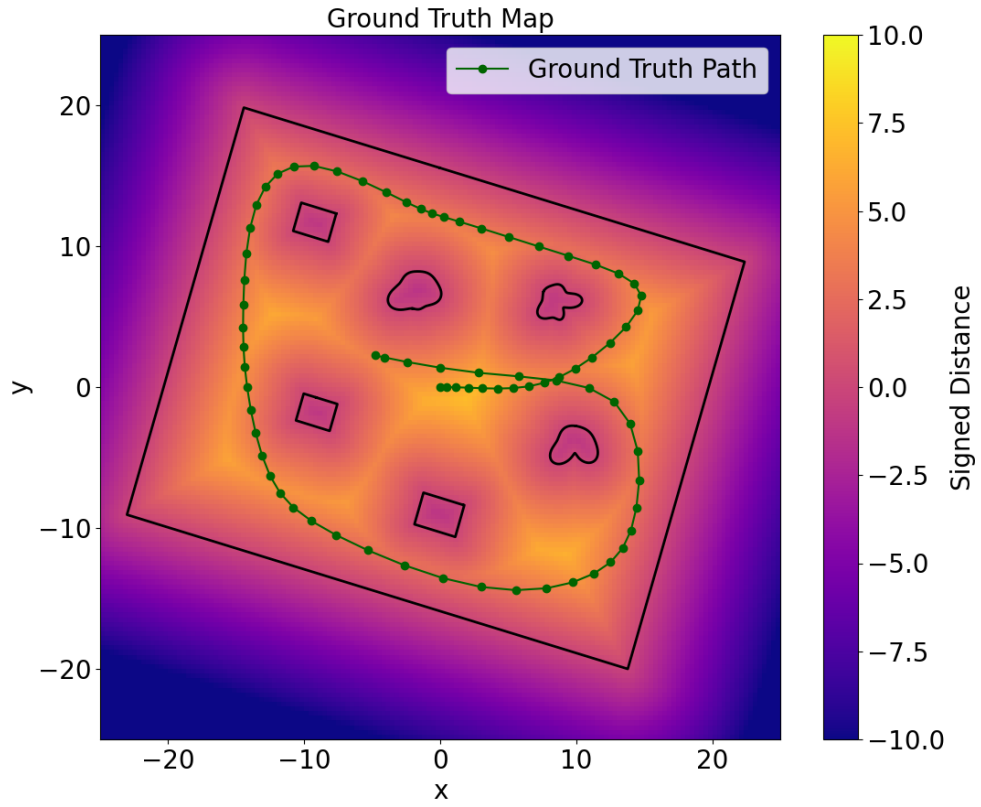
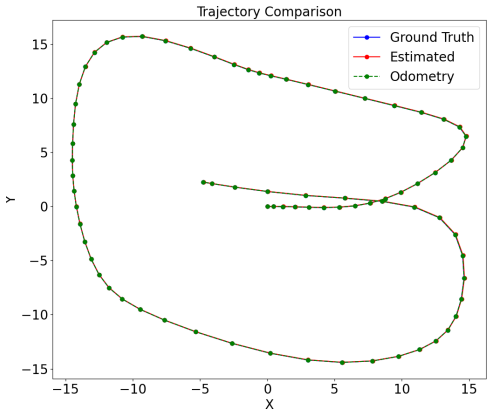


Figure 4.1: The environment defined over $\Omega = [-25, -25] \times [25, 25]$, illustrating the robot's trajectory and the environment from the simulated scan dataset.

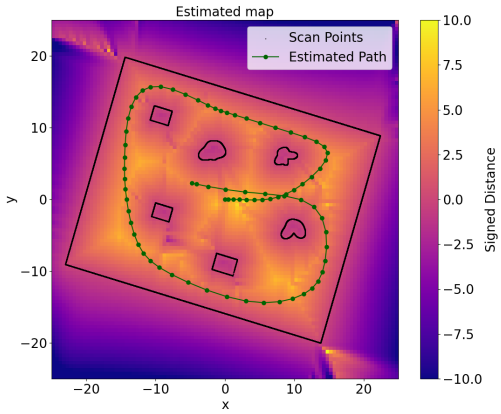
4.1.1 Experiment: Ground Truth Odometry

For this experiment, we use perfect odometry, which allows us to initialize the sensor poses with ground truth values. Consequently, the optimization primarily aims to recover an accurate continuous map while keeping the sensor poses near their initial ground truth values.

After optimization, the estimated poses remain nearly identical to the ground truth, with an average translation error of **0.0220** and an average rotation error of **0.00036** rad. Figure 4.2a shows an overlay of the ground truth, odometry, and optimized trajectories, confirming that the pose estimates are well preserved.



(a) Trajectory comparison for the simulated scan



(b) Reconstructed Map

Figure 4.2: Optimization result

Figure 4.1 shows the ground truth signed distance function, while Figure 4.2b displays the final reconstructed map. By comparing the signed distances of corresponding grid cell values between these two maps, we obtain an average error of around **0.7**.

Figure 4.3 presents a histogram of the error distribution across the map. Approximately 80% of the grid points exhibit an error below 1 unit, and 90% show an error below 3 units, indicating that most areas of the map are highly accurate.

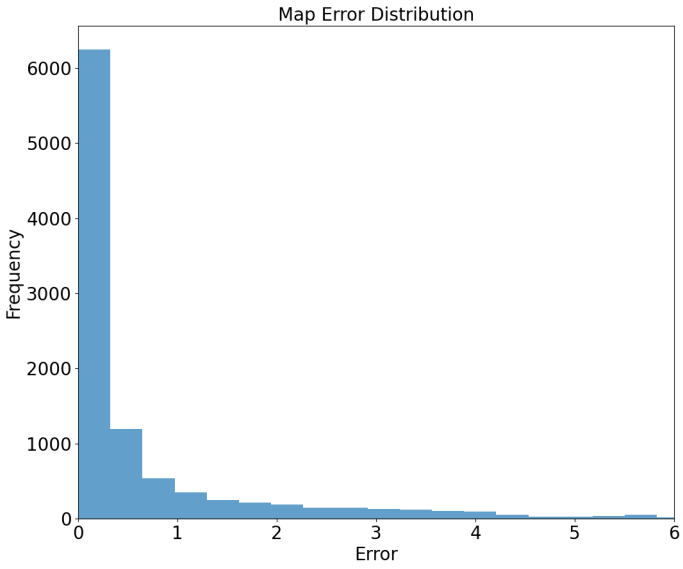
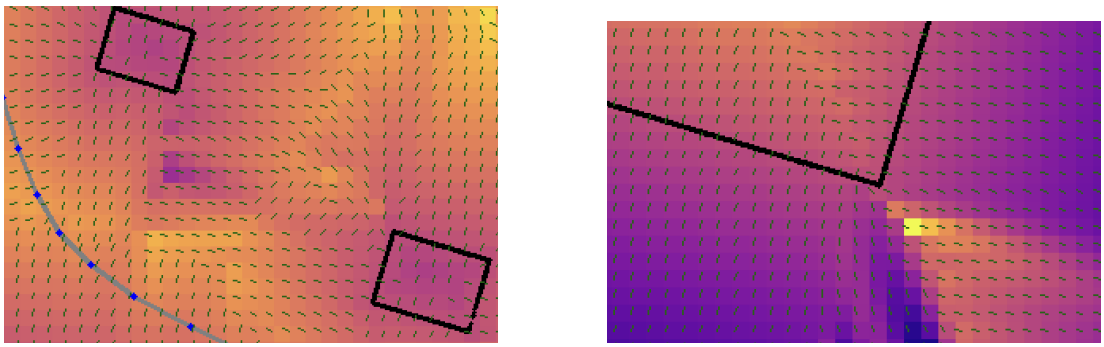


Figure 4.3: Error distribution across the map

Finally, Figure 4.4 highlights regions where inaccuracies in the estimated surface normals contribute to mapping errors. In Figure 4.4a, we can see problems at points that are equidistant to multiple objects, while Figure 4.4b shows issues near corners.

The green ticks represent the estimated normals at each grid point; areas where the normals change direction abruptly correlate with higher mapping errors. This is further confirmed by Figure B.1.



(a) Issues between objects.

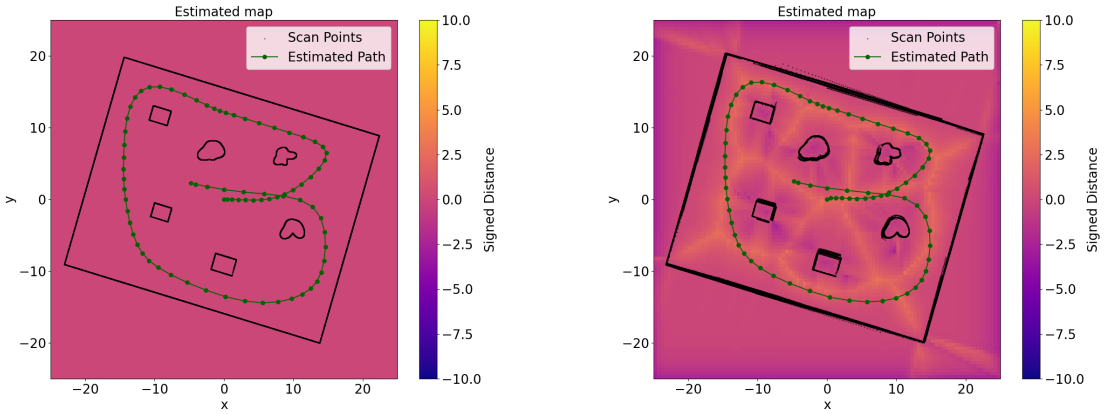
(b) Issues near corners.

Figure 4.4: Visualization of regions where inaccuracies in normal estimation lead to mapping errors. The black lines indicate objects, green ticks show the direction of the surface normals, the gray line shows the estimated path and the blue dots the estimated points at which the scans were taken.

4.1.2 Experiment: No Hallucinated Points

In this experiment, we assess the role of hallucinated points in the optimization process by running the algorithm with ground truth initialization for the poses but with the number of hallucinated points set to zero.

Figure 4.5 presents snapshots of the mapping process when hallucinated points are disabled. Initially, the scans are well aligned (Figure 4.5a). However, as the optimization proceeds, a divergence is observed: the scans are pushed apart (Figure 4.5b), and eventually, they realign (Figure 4.6a) except for frame 0, which remains fixed at $[0, 0, 0]$.

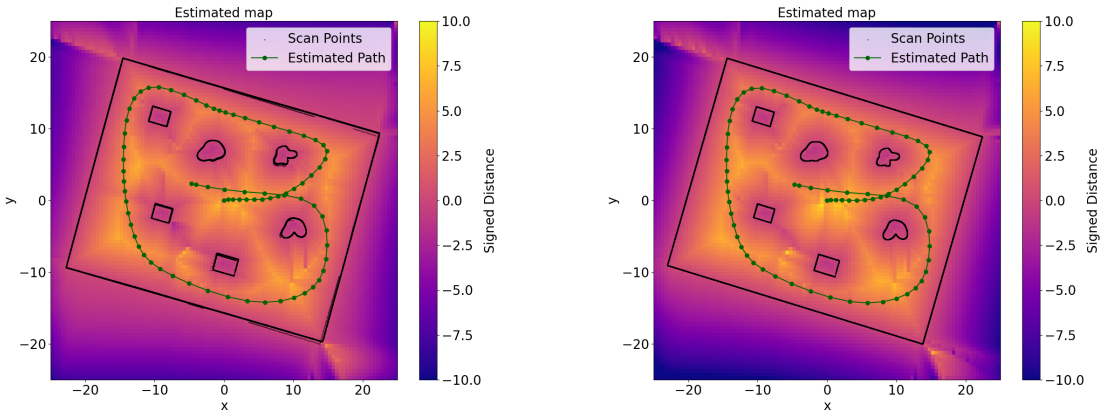


(a) Initial alignment

(b) Scans pushed apart

Figure 4.5: Snapshots from Experiment 2 illustrating the evolution of the map when hallucinated points are disabled.

If the optimization is allowed to run for additional iterations, the other frames gradually shift toward the fixed frame 0, as depicted in Figure 4.6b. Despite this eventual realignment, the overall trajectory remains less accurate when compared to the ground truth. While the global trajectory appears suboptimal, with an average absolute translation error of 0.2658 and an average rotation error of 0.01469 rad, the relative transformations between consecutive frames are precise. In particular, all relative transforms, except the one between frame 0 and frame 1, exhibit minimal error. As can be seen in Figure 4.7, the average relative translation error is only 0.0288 and the average relative rotation error is 0.00058 radians. This indicates that, although the lack of hallucinated points leads to a global misalignment (primarily due to the fixed first frame), the local consistency of the scan-to-scan registration remains nearly as accurate as when hallucinated points are used.



(a) Realigned (b) Global alignment

Figure 4.6: Map after additional optimization iterations, showing near alignment of all frames except the fixed frame 0.

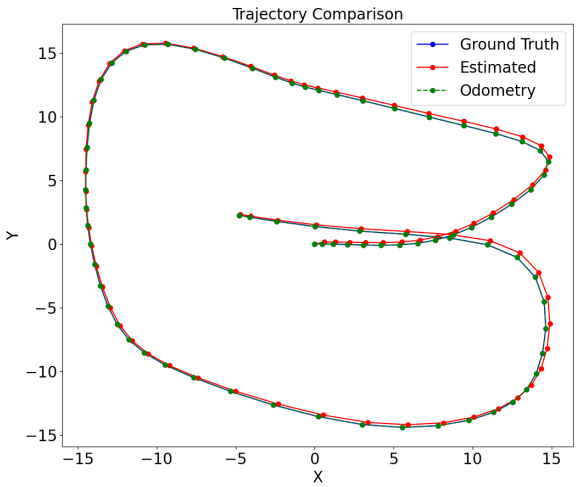


Figure 4.7: Trajectory comparison between the ground truth and the estimated trajectory in Experiment No Hallucinated Points.

4.1.3 Experiment: Noise

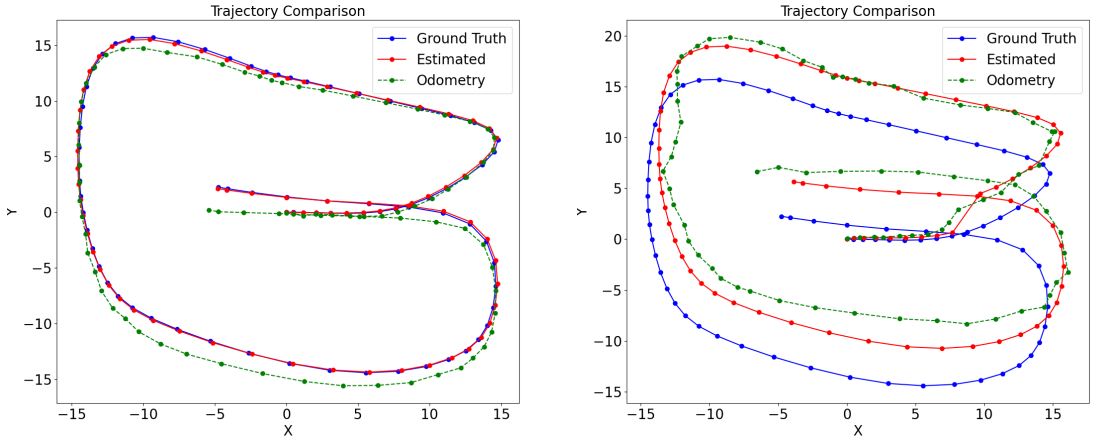
In this experiment, we evaluate the robustness of our SLAM system by introducing varying amounts of Gaussian noise to the ground truth odometry. Specifically, we add independent normally distributed noise with four different levels, ranging from $[0.1, 0.1, 0.01]$ (applied to dx, dy, and $d\theta$) up to $[0.3, 0.3, 0.03]$. This controlled degradation allows us to investigate how well the optimization process recovers the map and sensor transformations under adverse conditions.

Even with substantial noise, the algorithm is generally capable of reconstructing the map and recovering sensor transformations. However, similar to Experiment 4.1.2, the first frame remains fixed at $[0, 0, 0]$ and the other frames align with each other, but not with the first frame. This results in a relatively high transformation error for subsequent frames. As shown in Table 4.2, the relative errors remain nearly comparable to the noise-free case for noise levels up to $[0.2, 0.2, 0.02]$. In contrast, at the highest noise level of $[0.3, 0.3, 0.03]$, the optimization fails to converge to a reliable map or trajectory.

Condition	Avg Map Error	Avg Translation Error	Avg Rotation Error (rad)	Avg Rel Translation Error	Avg Rel Rotation Error (rad)
No noise	0.7082	0.0220	0.00037	0.0031	0.00014
No noise, no hallucinated points	1.4216	0.2657	0.01469	0.0287	0.00057
Noise [0.1, 0.1, 0.01]	0.7638	0.2104	0.01685	0.0290	0.00015
Noise [0.2, 0.2, 0.02]	0.9712	1.0036	0.00351	0.0254	0.00025
Noise [0.3, 0.3, 0.03]	1.0201	1.1203	0.03747	0.1188	0.00215

Table 4.2: Performance metrics under different noise conditions

Figure 4.8 compares the estimated trajectories under two noise conditions. In Figure 4.8a, the recovered trajectory aligns closely with the ground truth despite poor odometry inputs. Conversely, Figure 4.8b (severe noise $[0.3, 0.3, 0.03]$) shows that while most consecutive poses still exhibit reasonable local alignment, the overall path deviates significantly from the ground truth.

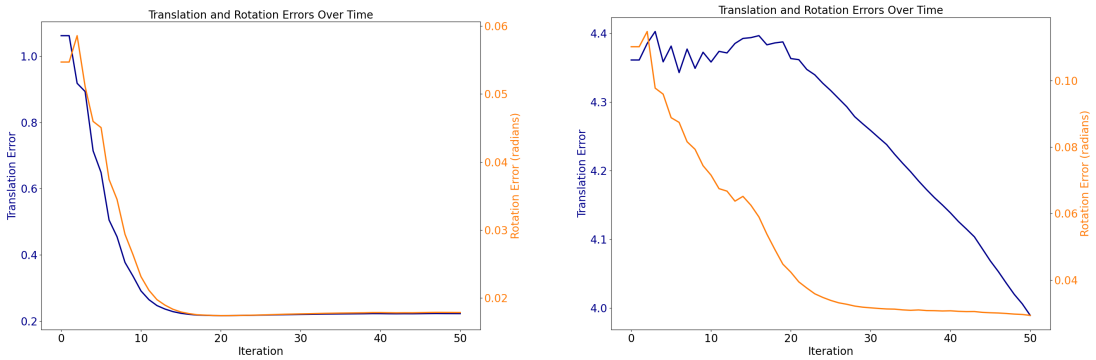


(a) Mild noise: $[0.1, 0.1, 0.01]$ (b) Severe noise: $[0.3, 0.3, 0.03]$

Figure 4.8: Estimated trajectories under different noise levels.

To further analyze the convergence behavior, Figure 4.9 presents two error evolution plots side by side. Figure 4.9a shows that with small amounts of noise, the transformation error decreases as the optimization proceeds and reaches a low level. In contrast,

Figure 4.9b focuses on the severe noise condition, where the error initially drops but then plateaus at a higher level. This plateau indicates that the optimization has become trapped in a local minimum, underscoring the sensitivity of our approach to noise.



(a) Slight noise: $[0.1, 0.1, 0.01]$ (b) Extreme noise: $[0.3, 0.3, 0.03]$

Figure 4.9: Evolution of the transformation error over time.

4.2 Intel Dataset Evaluation

The Intel Research Lab dataset [2] poses a particularly challenging scenario due to highly noisy odometry measurements and the lack of ground truth data. Consequently, our evaluation relies primarily on qualitative visual inspection.

Figure 4.10: Visualization of the Intel Research Lab environment [2]. Light blue lines show the path of the robot where the scans were taken.

4.2.1 Experiment: Batch Optimization

In this experiment, we initialize the sensor poses using the noisy odometry measurements and then perform a joint optimization over all 910 scans simultaneously.

The initial pose estimates, derived from the noisy odometry, are too inaccurate to yield a coherent result. Although we can observe local scan alignment, forming recognizable corridors, the overall map is highly inconsistent and distorted. This failure is evident in Figure 4.11, where the misaligned scans result in a cluttered and incoherent map. These results underscore the limitations of batch optimization when the input odometry is severely degraded.

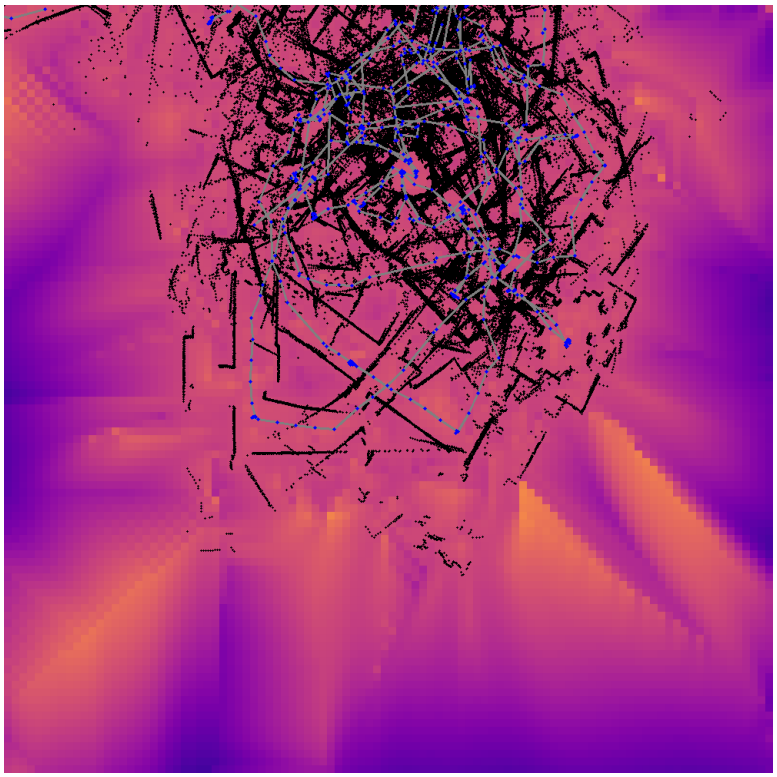


Figure 4.11: Result of batch optimization on the Intel dataset. Local scan alignments are visible. However, the overall map is highly inconsistent due to the poor initial pose estimates.

4.2.2 Experiment: Incremental Optimization

In this experiment, we employ an incremental optimization strategy on a subset of scans (scans 700–799) to assess how sequential refinement can correct noisy odometry. The process starts with only two scans, which are optimized for 10 iterations before adding the next scan. Each new scan is incorporated according to the relative odometry from the previously optimized scan. The goal is that the optimization corrects the odometry noise locally before integrating additional scans.

Otherwise using the hyperparameters from Table 4.1, the initial result (Figure 4.12) shows disjoint sets of aligned frames and fails to form a coherent map.

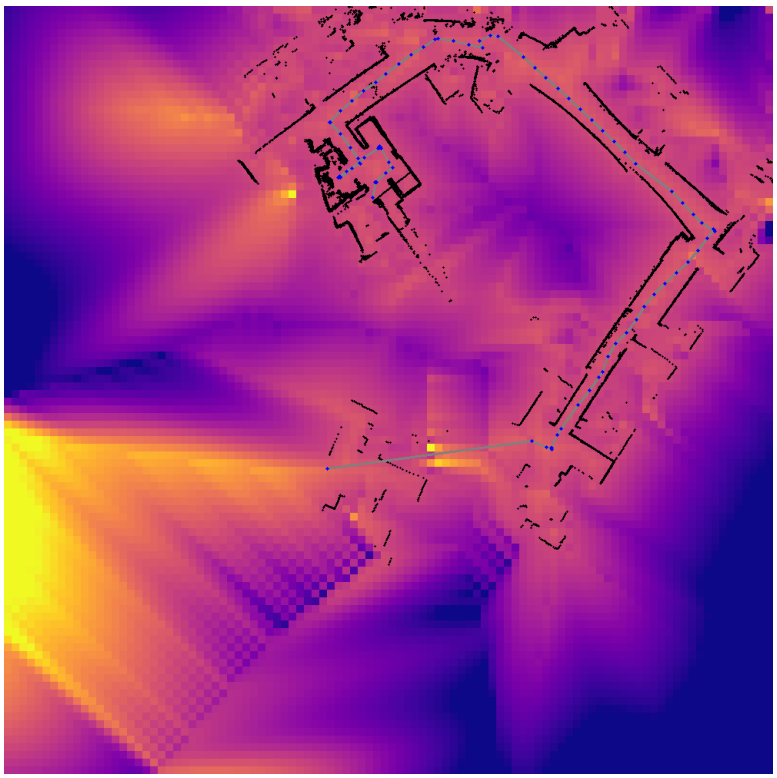


Figure 4.12: Initial result with default hyperparameters: Disjoint alignment due to poor initialization.

To explore the impact of the Eikonal residual, we first disabled it entirely. As seen in Figure 4.13, this leads to great scan alignment, but the map itself is not properly recovered because the SDF loses its regularization.

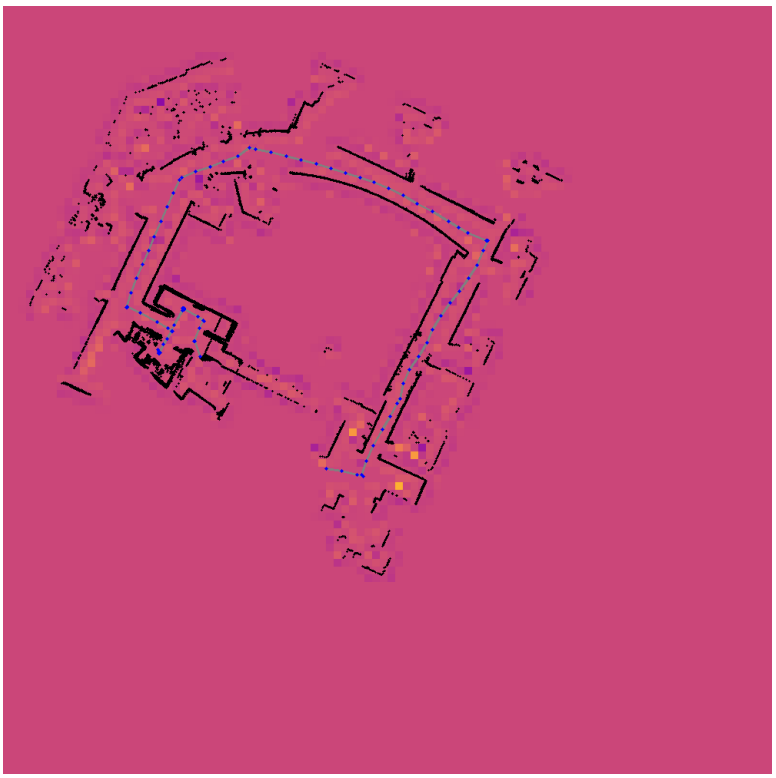
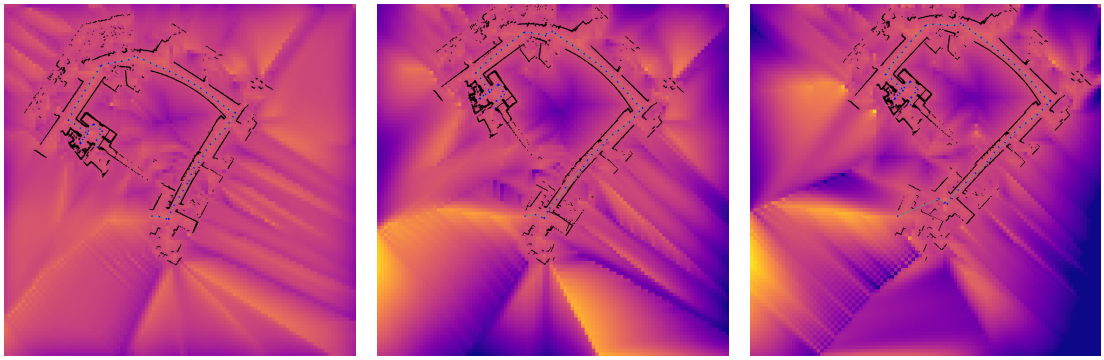


Figure 4.13: Result with the Eikonal residual disabled: Scans align well but the map is not recovered.

Next, we tested several values for the Eikonal weight w_e: 0.2, 0.5, and 0.8. Figure 4.14 shows a side-by-side comparison:



(a) $w_e = 0.2$ (b) $w_e = 0.5$ (c) $w_e = 0.8$

Figure 4.14: Comparison of incremental optimization results for different Eikonal weights. $w_e = 0.2$ achieves good alignment with some map recovery, while higher values yield poor alignment.

The results indicate that setting $w_e = 0.2$ yields satisfactory scan alignment and a modest improvement in the map. However, regardless of the w_e value tested, the final recovered map remains suboptimal. This issue primarily arises from the inherent ambiguity in distinguishing the interior from the exterior in scenes where objects are not closed,

which leads to inconsistencies. For example, while the central region in Figure 4.14b appears well defined, the periphery exhibits distortions resembling a jet-like structure, likely due to inaccuracies in the normal estimation process, as discussed in Section 4.1.1. This problem is further illustrated in Figure 4.15, which highlights the adverse impact of poor normal estimation on the map quality. We can see that the errors coincide with direction changes of the surface normals.

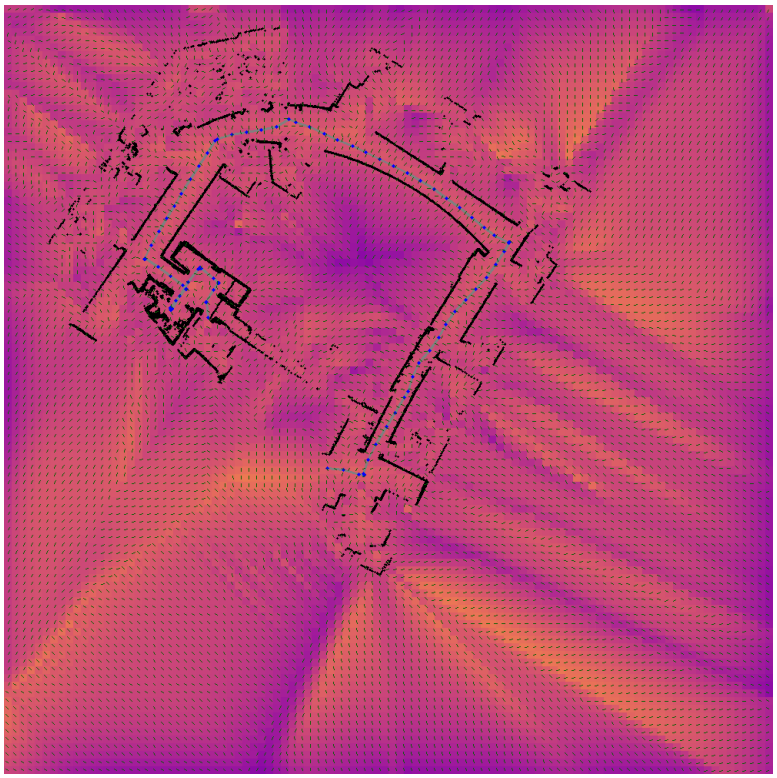


Figure 4.15: Visualization of mapping errors caused by poor normal estimation. The distortions along the map periphery highlight issues in accurately determining surface normals.

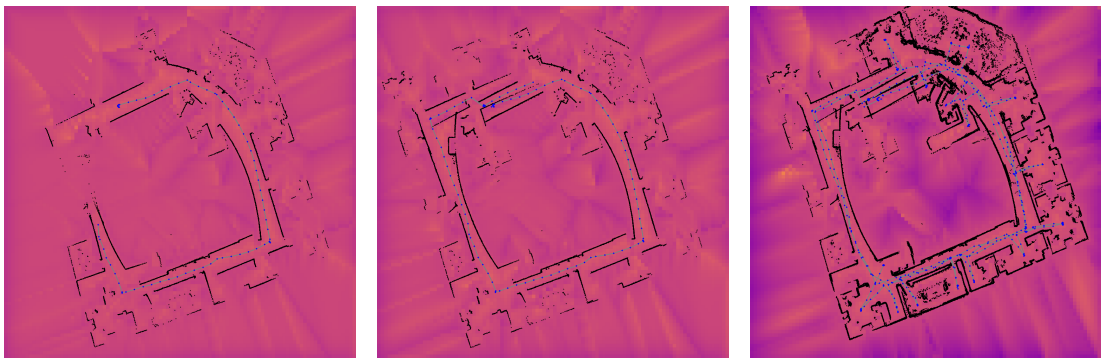
Overall, the incremental optimization experiment highlights a critical trade-off. While sequential refinement effectively reduces local odometry noise, the Eikonal constraint must be carefully tuned. An overly strong Eikonal term impedes scan alignment, whereas its complete removal compromises map recovery. These findings suggest that further investigation into adaptive weighting or alternative regularization methods is needed to robustly integrate scan data in challenging environments.

4.2.3 Experiment: Entire Dataset

Using the Eikonal weight $w_e = 0.2$ determined in Experiment 4.2.2, we apply our optimization framework to the entire Intel dataset. In this experiment, we vary the number of optimization iterations performed between the addition of each new frame. Specifically, we test configurations with 3, 5, and 10 iterations per new scan.

3 Iterations:

When using 3 iterations per new scan, the initial performance is promising. Locally the frames align well, however, as additional frames are incorporated, noticeable drift emerges. Figure 4.16 shows three stages of the evolution of the map: (a) the alignment before we reach the origin again, (b) the apparent drift after reaching our starting point again, and (c) the accumulated drift resulting in inaccurate alignment and thick walls. This highlights that without explicit loop closure the algorithm struggles to achieve global consistency.



(a) Alignment before reaching the origin

(b) Drift becomes apparent

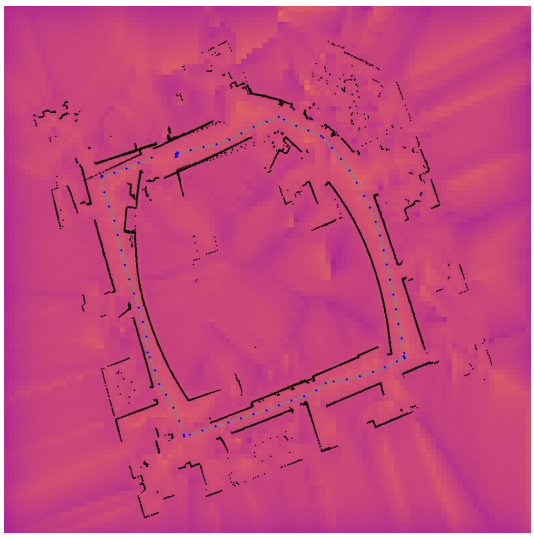
(c) Accumulated drift

Figure 4.16: Map evolution with 3 iterations per new scan.

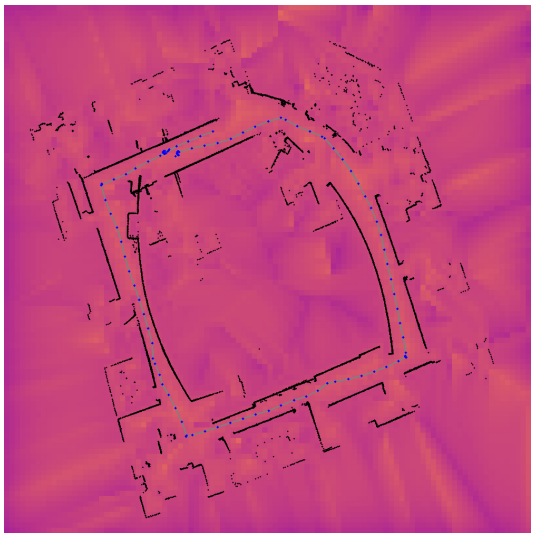
5 Iterations:

Increasing the iterations to 5 per new scan reduced the drift compared to the 3-iteration configuration. Figure 4.17 shows three stages: (a) when reaching the starting point again we see drift, (b) after a couple more iterations all scans realign to reduce the drift, (c) the scans are almost globally aligned, and (d) zooms in to the top left corner to highlight the misalignment. The two green circles show that even though it initially seems that the scans are now globally aligned, that there is a small shift along the corridor leading to the door being covered by a wall and the the walls not quite aligning with each other.

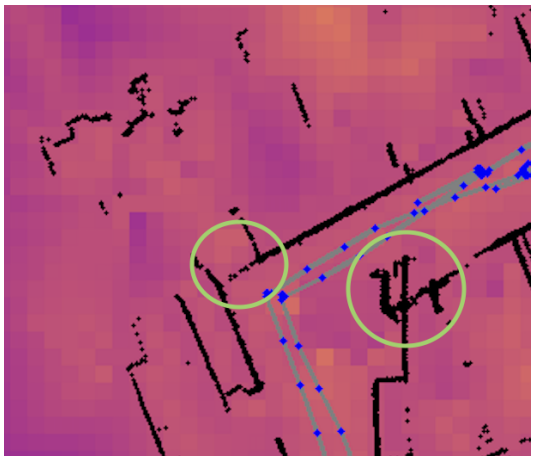
(a) Initial alignment



(b) Reduced drift



(c) Improved alignment



(d) Misalignment

Figure 4.17: Map evolution with 5 iterations per new scan.

10 Iterations:

Increasing the iterations to 10 per new frame significantly improves drift correction, enabling the algorithm to achieve global consistency even without an explicit loop closure mechanism. Figure 4.18 illustrates the alignment, effectively eliminating drift.

A closer inspection is provided by Figure 4.19, comparing the initial state with notable drift (Figure 4.19a) to a zoomed-in view showcasing improved alignment of structural features such as doors and walls (Figure 4.19b). This is a marked improvement compared to the misalignment previously observed in Figure 4.17d.

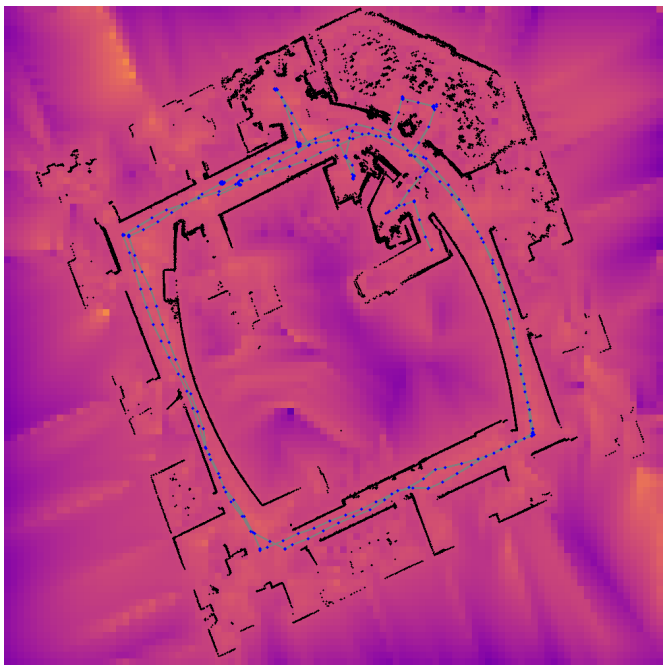
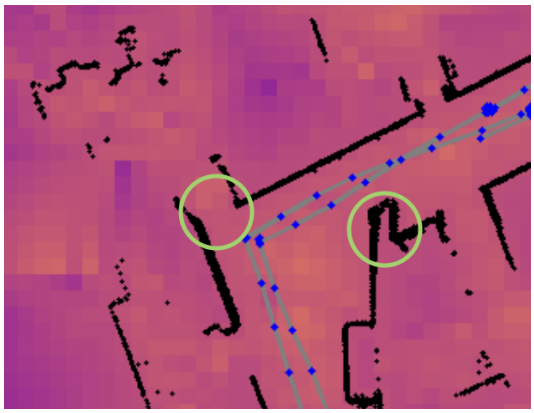


Figure 4.18: Map alignment with 10 iterations per new frame.

(a) Initial alignment



(b) Improved alignment of doors and walls

Figure 4.19: Comparison of map alignment before and after optimization.

Chapter 5

Discussion & Analysis

In this dissertation, we have presented a proof-of-concept SLAM system that jointly optimizes a continuous signed distance function and sensor poses. Our experimental evaluation on both simulated data and the challenging Intel Research Lab dataset demonstrates promising local alignment and relative transformation accuracy. However, several issues remain before the approach can compete with state-of-the-art SLAM systems.

One significant challenge is the difficulty of comparing our method to existing work. Many established frameworks are implemented in ROS and have been refined over years of development, while our implementation in C++ serves primarily as a proof of concept. A thorough quantitative comparison on standard benchmarks would require additional work beyond the scope of this dissertation.

We will now try and answer the questions presented in the introduction:

5.1 Is a unified optimization formulation promising?

Our experiments reveal that the framework demonstrates strong robustness to local odometry noise, and notably, under certain conditions, it can achieve global consistency without explicitly employing a loop closure mechanism. Specifically, results from the Intel dataset indicate that with sufficient optimization iterations per frame, global drift can be significantly reduced or even eliminated entirely. While integrating loop closure into the nonlinear least-squares framework remains a potential enhancement to further mitigate drift in general cases, our findings show evidence that careful optimization alone can, in some scenarios, effectively replace the need for explicit loop closure detection.

In certain cases, the joint optimization of the map and sensor poses can have a counterproductive effect, pushing newly added poses further away from their true locations before they align with the rest of the frames, as observed in Experiment 4.2.2. One potential remedy is to “lock” the map for several iterations before continuing with joint optimization, thereby stabilizing the pose estimates and preventing premature, erroneous adjustments.

The role of hallucinated points remains ambiguous. While they appear to help compensate for noisy odometry and improve local alignment, they also account for a large portion of the computational load. Our experiments suggest that their contribution is sensitive to hyperparameter settings, and further research is needed to clarify when their inclusion is truly beneficial.

Furthermore, the map representation poses its own challenges. The map size is defined statically at the beginning of the optimization, resulting in unnecessary computational overhead since only a fraction of the map may be actively used at the beginning. A dynamic map that adapts to the region of interest, or a non-uniform resolution strategy, where high resolution is maintained in critical areas (e.g., along corridors) and lower resolution elsewhere, could yield both computational savings and improved map fidelity.

5.2 Can a continuous SDF yield accurate map reconstructions?

In the controlled environment of simulated scans, the Eikonal residuals largely function as intended and yield an accurate map. However, in the more complex and noisy Intel dataset, the Eikonal constraint fails to produce the desired results. This shortcoming appears to stem from inadequate surface normal estimation at grid points, which undermines the consistency of the gradient and degrades the map quality. Although the current approach successfully determines the correct polarity, that is, distinguishing which side of the surface is positive or negative, integrating additional cues, such as the vector toward the closest point, may improve normal estimation and enhance overall performance.

5.3 Is the overall approach computationally feasible?

Another critical consideration is computational efficiency. Currently, optimizing the Intel dataset, which consists of 910 frames and approximately 155,648 scan points, requires approximately 10 seconds per iteration without hallucinated points. Introducing hallucinated points increases computational time substantially, resulting in approximately 16 seconds per iteration with two hallucinated points, 23 seconds with four, and up to 30 seconds with six. Achieving real-time performance would necessitate roughly a 500-fold reduction in computational time, highlighting the need for substantial improvements both in algorithmic design and implementation efficiency. Potential strategies to mitigate this computational challenge include employing parallel computation techniques or adopting hierarchical optimization methods that separately address local and global adjustments.

5.4 Summary

In summary, while the proposed SDF-based SLAM framework shows encouraging results as a proof of concept, several aspects need further refinement. The interplay

between hallucinated points, Eikonal residuals, and pose optimization requires deeper investigation. Future work should explore adaptive strategies for map resolution and dynamic partitioning of the optimization problem to enhance both accuracy and computational efficiency. Despite these limitations, the current findings provide a solid foundation for developing a more robust, real-time SLAM system in subsequent research.

Chapter 6

Conclusion & Future Work

6.1 Summary of Findings

Our experiments have demonstrated that a unified optimization formulation, which jointly refines sensor poses and a continuous SDF map, shows considerable promise. In controlled, simulated environments, the continuous SDF can yield highly accurate map reconstructions, even in the presence of noisy odometry. However, when applied to more complex real-world scenarios (as exemplified by the Intel dataset), challenges such as global drift, limitations in the Eikonal residual enforcement, and inaccuracies in normal estimation become apparent. While the approach is computationally feasible as a proof of concept, achieving real-time performance remains an open challenge.

6.2 Contributions

This work makes several key contributions:

- We introduce a novel unified optimization framework that concurrently estimates sensor poses and a continuous SDF map, addressing the research question of whether a unified formulation is promising.
- Our approach demonstrates that a continuous SDF representation can produce accurate map reconstructions under controlled conditions, offering a compelling alternative to traditional discrete mapping methods.
- We provide an in-depth analysis of the computational feasibility of joint optimization, highlighting both the strengths and current limitations of our method.
- Our incremental optimization strategy offers valuable insights into the trade-offs between local scan alignment and global map consistency.

6.3 Future Work

Future research should focus on several key areas to enhance the robustness and efficiency of the framework:

- **Loop Closure:** Integrating a loop closure mechanism into the nonlinear least-squares formulation is essential for mitigating global drift and ensuring consistent global map alignment.
- **Multi-Stage Optimization:** Investigating a multi-step optimization strategy—such as temporarily fixing the map while incorporating new frames and then performing joint re-optimization—may help prevent Eikonal residuals from pushing new pose estimates off course.
- **Performance Optimization:** Significant improvements in computational efficiency are required. Approaches such as processing multiple scans concurrently (e.g., adding k scans at once) and optimizing the sparse matrix operations are necessary to move towards real-time performance.
- **Normal Estimation:** The current method for surface normal estimation is inadequate, particularly in complex, real-world scenarios. Future work should focus on refining normal estimation, potentially by integrating additional geometric cues (such as the vector towards the closest point) to enhance the accuracy of the SDF reconstruction.
- **Extension to 3D:** Although the methodology is presented in a 2D framework, it naturally extends to three dimensions. In 3D, bilinear interpolation is replaced by trilinear interpolation and the finite difference approximations are extended to account for the additional dimension. Rigid-body transformations are similarly defined in 3D, while the overall optimization framework remains consistent. Nevertheless, further experiments are necessary to assess any new challenges and to address the increased computational complexity that comes with extending the approach to the third dimension.

Bibliography

[1] Bilinear Interpolation Calculator — omnicalculator.com. https://www.omnicalculator.com/math/bilinear-interpolation. [Accessed 27-03-2025].

[2] URL http://ais.informatik.uni-freiburg.de/slamevaluation/datasets.php.

[3] URL https://www.ipb.uni-bonn.de/html/teaching/photo12-2021/2021-pho2-15-slam-intro.pptx.pdf.

[4] URL https://vocal.media/gamers/how-slam-technology-revolutionizes-augmente

[5] Tim Bailey and Hugh F. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics & Automation Magazine*, 13:108–117, 2006. URL https://api.semanticscholar.org/CorpusID:6542016.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

[7] Berta Bescos, Jose M. Facil, Javier Civera, and Jose Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, October 2018. ISSN 2377-3774. doi: 10.1109/lra.2018.2860039. URL http://dx.doi.org/10.1109/LRA.2018.2860039.

[8] E.H. Bruns. *Das Eikonal, von Heinrich Bruns*. S. Hirzel, 1895. URL https://books.google.co.uk/books?id=kjDytgAACAAJ.

[9] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose 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, December 2016. ISSN 1941-0468. doi: 10.1109/tro.2016.2624754. URL http://dx.doi.org/10.1109/TRO.2016.2624754.

[10] Gadiel Sznaier Camps, Robert Dyro, Marco Pavone, and Mac Schwager. Learning deep sdf maps online for robot navigation and exploration, 2022. URL https://arxiv.org/abs/2207.10782.

[11] Jeremy Cohen. Introduction to loop closure detection in slam, Sep 2023. URL https://www.thinkautonomous.ai/blog/loop-closure/.

[12] Aron J. Cooper. A comparison of data association techniques for simultaneous localization and mapping. 2005. URL https://api.semanticscholar.org/CorpusID:109358569.

[13] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 1996. URL https://api.semanticscholar.org/CorpusID:260380609.

[14] Frank Dellaert and GTSAM Contributors. borglab/gtsam, May 2022. URL https://github.com/borglab/gtsam).

[15] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25:1181 – 1203, 2006. URL https://api.semanticscholar.org/CorpusID:667030.

[16] Frank Dellaert and Michael Kaess. *Factor Graphs for Robot Perception*. Foundations and Trends in Robotics, Vol. 6, 2017. URL http://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf.

[17] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001. doi: 10.1109/70.938381.

[18] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006. doi: 10.1109/MRA.2006.1638022.

[19] Hugh F. Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13:99–110, 2006. URL https://api.semanticscholar.org/CorpusID:8061430.

[20] Alberto Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. 1, 03 2013. doi: 10.48550/arXiv.1304.1098.

[21] Joscha-David Fossel, Karl Tuyls, and Jürgen Sturm. 2d-sdf-slam: A signed distance function based slam frontend for laser scanners. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1949–1955, 2015. doi: 10.1109/IROS.2015.7353633.

[22] Giorgio Grisetti, C. Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2432–2437, 2005. URL https://api.semanticscholar.org/CorpusID:6221209.

[23] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes, 2020. URL https://arxiv.org/abs/2002.10099.

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

[25] Pawel Kicman, Peter Silson, Antonios Tsourdos, and Al Savvaris. Cartographer - a terrain mapping and landmark-based localization system for small robots. 03 2011. ISBN 978-1-60086-944-0. doi: 10.2514/6.2011-1513.

[26] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007. doi: 10.1109/ISMAR.2007.4538852.

[27] Philipp Koch, Stefan May, Michael Schmidpeter, M. Kühn, Christian Pfitzner, Christian Merkl, Rainer Koch, Martin Fees, Jon Martin, and A. Nüchter. Multi-robot localization and mapping based on signed distance functions. *Proceedings - 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2015*, pages 77–82, 05 2015. doi: 10.1109/ICARSC.2015.18.

[28] Aravindh Krishnamoorthy and Deepak Menon. Matrix inversion using cholesky decomposition. In *2013 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 70–72, 2013.

[29] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. doi: 10.1109/ICRA.2011.5979949.

[30] Tin Lai. A review on visual-slam: Advancements from geometric modelling to learning-based semantic scene understanding, 2022. URL https://arxiv.org/abs/2209.05222.

[31] Tony Lindeberg. *Scale Invariant Feature Transform*, volume 7. 05 2012. doi: 10.4249/scholarpedia.10491.

[32] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11 (2):431–441, 1963. ISSN 03684245. URL http://www.jstor.org/stable/2098941.

[33] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers Geosciences*, 19(3):303–342, 1993. ISSN 0098-3004. doi: https://doi.org/10.1016/0098-3004(93)90090-R. URL https://www.sciencedirect.com/science/article/pii/009830049390090R.

[34] Michael Milford and Ashley George. *Featureless Visual Processing for SLAM in Changing Outdoor Environments*, volume 92, pages 569–583. 01 2014. ISBN 978-3-642-40685-0. doi: 10.1007/978-3-642-40686-7_38.

[35] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5): 1147–1163, October 2015. ISSN 1941-0468. doi: 10.1109/tro.2015.2463671. URL http://dx.doi.org/10.1109/TRO.2015.2463671.

[36] Richard Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. pages 127–136, 10 2011. doi: 10.1109/ISMAR.2011.6092378.

[37] 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, 2022. URL https://arxiv.org/abs/2204.02296.

[38] 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*, 40:4045–4064, 2024. ISSN 1941-0468. doi: 10.1109/tro.2024.3422055. URL http://dx.doi.org/10.1109/TRO.2024.3422055.

[39] Zhexi Peng, Tianjia Shao, Liu Yong, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. 2024.

[40] Enrique Fernandez Perdomo. Test and evaluation of the fastslam algorithmin a mobile robot. 2009. URL https://api.semanticscholar.org/CorpusID:134648979.

[41] Julio A. Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A. Castellanos. A survey on active simultaneous localization and mapping: State of the art and new frontiers, 2023. URL https://arxiv.org/abs/2207.00254.

[42] Pengtao Qu, Chen Su, Hang Wu, Xinxi Xu, Sheng Gao, and Xiuguo Zhao. Mapping performance comparison of 2d slam algorithms based on different sensor combinations. *Journal of Physics: Conference Series*, 2024:012056, 09 2021. doi: 10.1088/1742-6596/2024/1/012056.

[43] 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, 2020. doi: 10.1109/LRA.2019.2953859.

[44] 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. doi: 10.1177/02783649211056674. URL https://doi.org/10.1177/02783649211056674.

[45] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.

[46] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *ICRA*. IEEE, 2011. URL http://dblp.uni-trier.de/db/conf/icra/icra2011.html#RusuC11.

[47] Shahrizal Saat, WN Rashid, MZM Tumari, and Muhammad Salihin Saealal. Hectorslam 2d mapping for simultaneous localization and mapping (slam). *Journal of Physics: Conference Series*, 1529:042032, 04 2020. doi: 10.1088/1742-6596/1529/4/042032.

[48] Sergei Solonets, Daniil Sinitsyn, Lukas Von Stumberg, Nikita Araslanov, and Daniel Cremers. An analytical solution to gauss-newton loss for direct image alignment. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mE52zURNGc.

[49] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time, 2021. URL https://arxiv.org/abs/2103.12352.

[50] Sebastian Thrun. Robotic mapping: a survey. 2003. URL https://api.semanticscholar.org/CorpusID:14633188.

[51] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *I. J. Robotic Res.*, 25:403–429, 05 2006. doi: 10.1177/0278364906065387.

[52] Sebastian Thrun, Daphne Koller, Zoubin Ghahramani, Hugh F. Durrant-Whyte, and A. Ng. Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In *Workshop on the Algorithmic Foundations of Robotics*, 2004. URL https://api.semanticscholar.org/CorpusID:7428906.

[53] Sebastian Thrun, Michael Montemerlo, Daphne Koller, Ben Wegbreit, Juan Nieto, and Eduardo Nebot. Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data. *Journal of Machine Learning Research*, 4, 05 2004.

[54] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[55] Sebastian Thrun, Michael Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *J. Field Robotics*, 23:661–692, 01 2006.

[56] Matthew R. Walter, Ryan M. Eustice, and John J. Leonard. Exactly sparse extended information filters for feature-based slam. *The International Journal of Robotics Research*, 26:335 – 359, 2007. URL https://api.semanticscholar.org/CorpusID:1666309.

[57] Jun Wang, Li Zhang, Yanjun Huang, Jian Zhao, and Francesco Bella. Safety of autonomous vehicles. *Journal of Advanced Transportation*, 2020:1–13, 10 2020. doi: 10.1155/2020/8867757.

[58] Jiali Xu, Di Wang, Maosheng Liao, and Wenshuai Shen. Research of cartographer graph optimization algorithm based on indoor mobile robot. *Journal of Physics: Conference Series*, 1651:012120, 11 2020. doi: 10.1088/1742-6596/1651/1/012120.

[59] Xianghan Yan, Weichao Guo, Yanhong Wang, and Shuofei Li. Application of visual slam technology in military intelligent unmanned systems. In *Proceedings of the 2024 4th International Conference on Artificial Intelligence, Big Data and Algorithms*, CAIBDA '24, page 206–211, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400710247. doi: 10.1145/3690407.3690442. URL https://doi.org/10.1145/3690407.3690442.

[60] Xianghan Yan, Weichao Guo, Yanhong Wang, and Shuofei Li. Application of visual slam technology in military intelligent unmanned systems. pages 206–211, 10 2024. doi: 10.1145/3690407.3690442.

[61] Zhiyuan Yang, Nabila Naz, Pengcheng Liu, and M. Huda. *Evaluation of SLAM Algorithms for Search and Rescue Applications*, pages 114–125. 09 2023. ISBN 978-3-031-43359-7. doi: 10.1007/978-3-031-43360-3_10.

[62] Liang Zhao, Yingyu Wang, and Shoudong Huang. Occupancy-slam: Simultaneously optimizing robot poses and continuous occupancy map. In *Robotics: Science and Systems XVIII*, RSS2022. Robotics: Science and Systems Foundation, June 2022. doi: 10.15607/rss.2022.xviii.003. URL http://dx.doi.org/10.15607/RSS.2022.XVIII.003.

Appendix A

Analysis of Methodology

A.1 Eikonal Equation

Gropp et al. [23] established that the signed distance function must satisfy the Eikonal equation:

$$|\nabla D| = 1. \tag{A.1}$$

A.1.1 Projecting the derivative vector

Let us consider a scenario where two infinite walls define the axes of our coordinate system, restricting our analysis to the first quadrant. The signed distance function (SDF) in this setting is given by:

$$d(x,y) = \min(x,y).$$

A.1.1.1 Gradient Computation

The derivative of $d(x,y)$ is determined as follows:

- If $x < y$, then $d(x,y) = x$, yielding the partial derivatives:

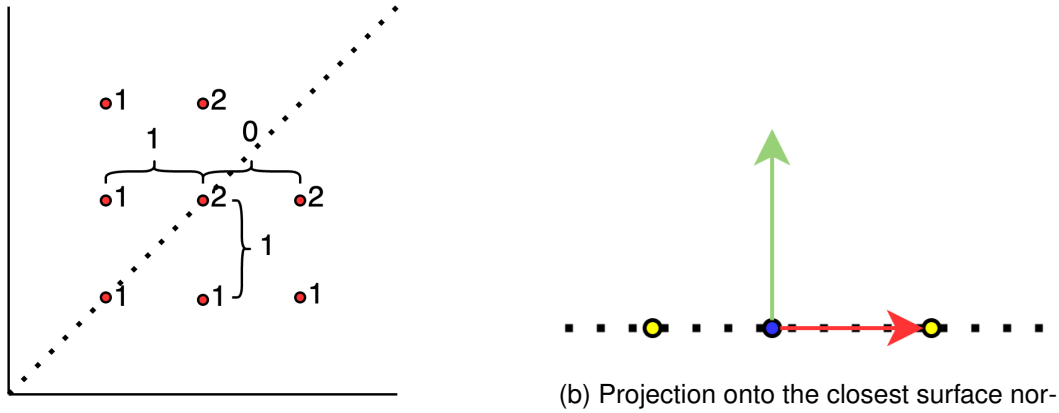
$$\frac{\partial d}{\partial x} = 1, \quad \frac{\partial d}{\partial y} = 0.$$

- If $x > y$, then $d(x,y) = y$, resulting in:

$$\frac{\partial d}{\partial x} = 0, \quad \frac{\partial d}{\partial y} = 1.$$

At the diagonal $x = y$, the gradient is not uniquely defined. The possible gradients form a convex combination of the limiting gradients:

$$(1,0) \quad (\text{from } x < y), \quad (0,1) \quad (\text{from } x > y).$$



(a) Illustration of gradient estimation error

(b) Projection onto the closest surface normal

Figure A.1: Handling undefined gradients at discontinuities

Thus, the set of possible gradients at $x = y$ is given by:

$$\{(\alpha, 1-\alpha) \mid 0 \leq \alpha \leq 1\}.$$

A.1.1.2 Numerical Estimation and Issues

Using forward differencing for numerical gradient estimation may lead to an incorrect gradient. Figure A.1a illustrates this issue, where the computed gradient at a grid point along the diagonal results in $(1,1)$, violating the Eikonal equation. The numbers next to the red dots represent the computed SDF values, while the brackets highlight the estimated gradients.

To address this issue, we can constrain the computed gradient by projecting it onto one of the subgradients. Specifically, we propose selecting a direction that points towards the closest surface and projecting the estimated gradient onto this direction. For points where $x \neq y$, this projection does not affect the gradient magnitude. However, at locations along the diagonal, it enforces a well-defined gradient with unit magnitude, satisfying the Eikonal equation.

Consider the example of $(x,y) = (2,2)$. Here, we can choose between the vectors $(-1,0)$ and $(0,-1)$. Projecting the computed gradient onto either of these vectors ensures a magnitude of 1, aligning with the Eikonal constraint.

A.1.1.3 Generalized Approach to Finding the Closest Surface Normal

In a more general setting, the closest surface direction is not always known a priori. There are two practical ways to approximate it:

1. **Nearest Scan Point Approximation:** The simplest approach is to use the vector to the closest scan point. However, this can lead to significant inaccuracies. For instance, if scan points are uniformly spaced at integer coordinates such as $(0,0)$, $(0,1)$, etc., then a point at $(0,0.6)$ would identify $(0,1)$ as the closest scan point, yielding a direction of $(0,1)$. This is incorrect if the true surface normal at that location is $(1,0)$, which is

orthogonal to the computed direction. This issue is visualized in Figure A.1b, where yellow points represent scan points, the blue point is a grid node, the red vector indicates the direction to the closest scan point, and the green vector represents the correct surface normal.

2. **Surface Normal Estimation from Scan Points:** Instead of relying solely on the closest scan point, we can estimate the surface normal at each scan point and use the normal of the nearest scan point for gradient correction. In Figure A.1b, the estimated surface normal at each scan point is shown as a green arrow. By using this estimated normal at the closest scan point, we obtain a more accurate approximation of the true surface normal at the query point.

Since surface normals are inherently ambiguous (they can point in two opposing directions), we resolve this by consistently orienting normals toward the scan origin. This choice provides a secondary benefit: in regions in front of the scanned surface, the normals naturally point outward, while behind the surface, they point inward. This ensures the signed distance function correctly represents positive and negative distances, maintaining consistency with the Eikonal equation.

A.1.1.4 Experiment: Projecting vs. Not Projecting

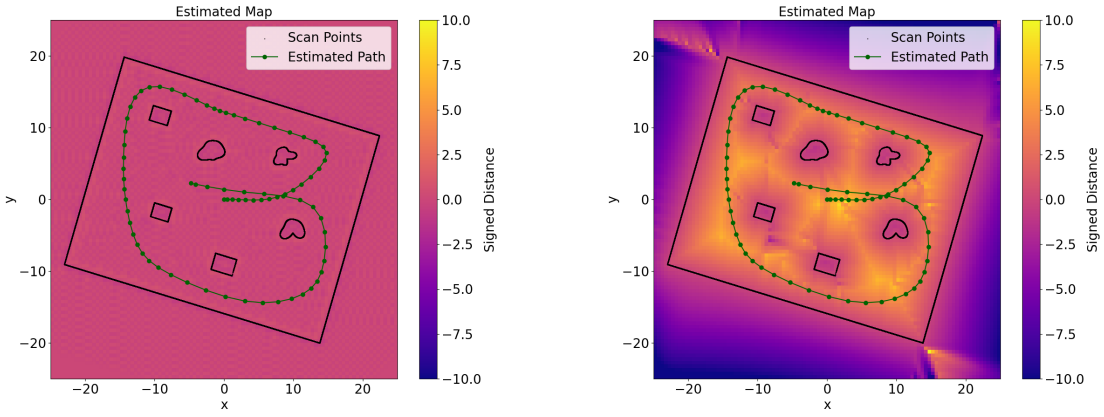
To test our hypothesis regarding the effect of projecting the derivative onto the estimated surface normal, we conducted an experiment using the simulated dataset described in Section 4.1. In this experiment, we optimized the map for 25 iterations under two conditions: one in which the derivative is projected onto the surface normal, and one in which no projection is applied.

Figure A.2 shows the results. As seen in Figure A.2a, when no projection is applied, the map fails to recover correctly and exhibits an alternating pattern. This behavior is not entirely unexpected—using forward differencing, a sequence such as 0, 1, 0, 1 will produce a derivative magnitude of 1 everywhere (with central differencing the derivative would be 0, yet similar behavior still occurs). In contrast, Figure A.2b demonstrates that projecting the derivative onto the estimated surface normal leads to a map that is recovered with considerably greater fidelity, although additional iterations may be required to further refine the reconstruction.

A.1.2 Differencing Scheme

However, since we represent the SDF on a discrete grid, we must approximate the gradient numerically. Various finite differencing schemes offer distinct advantages and drawbacks. The simplest approach, both in terms of implementation and computational efficiency, is the forward difference method, which we adopt in this paper. While central differencing provides higher accuracy, it introduces an intriguing phenomenon.

For our analysis, we restrict ourselves to a one-dimensional setting and consider the absolute distance to the surface. The only known information is that grid point 4 corresponds to the surface, meaning the distance at this point must be zero. We then compute the Eikonal residuals and apply least squares optimization.



(a) Without Projection

(b) With Projection

Figure A.2: Comparison of map recovery after 25 iterations. The left image, obtained without projecting the derivative, shows an alternating pattern that hinders accurate map reconstruction. In contrast, the right image, with derivative projection applied, demonstrates a more faithful recovery of the environment.

The left side of Figure A.3 illustrates that forward differencing correctly reconstructs the SDF. However, the right side reveals an alternating pattern arising from the central difference scheme. This phenomenon occurs because the central difference at index i depends on values at $i+1$ and $i-1$, meaning that adjacent grid points do not interact directly. Instead, they only influence their second neighbors. Given that we only specify information at grid point 4 in this example, the computed SDF directly or indirectly constrains only grid points 0, 2, 6, and 8. The remaining grid points (1, 3, 5, and 7) are unconstrained, allowing them to assume arbitrarily large or small values while still fitting a straight-line solution. The specific line they conform to depends on their initial values, and this effect is highlighted in orange.

A.1.2.1 Experiment: Central vs Forward Differencing

To investigate whether the observed behavior is merely a theoretical edge case or also manifests in more complex environments, we conducted an experiment using the simulated dataset described in Section 4.1.

Figure A.6 shows the maps produced after running the optimization for 100 iterations with all settings held constant except for the type of derivative approximation used. A qualitative inspection reveals that the forward difference method (Figure A.4a) yields a noticeably smoother map, while the central difference method (Figure A.4b) exhibits the alternating behavior described previously. This can be further observed in Figure A.5. Quantitatively, the average map error is 0.70 for the forward difference approach compared to 0.83 for the central difference method, indicating a significant performance improvement when using forward differences.

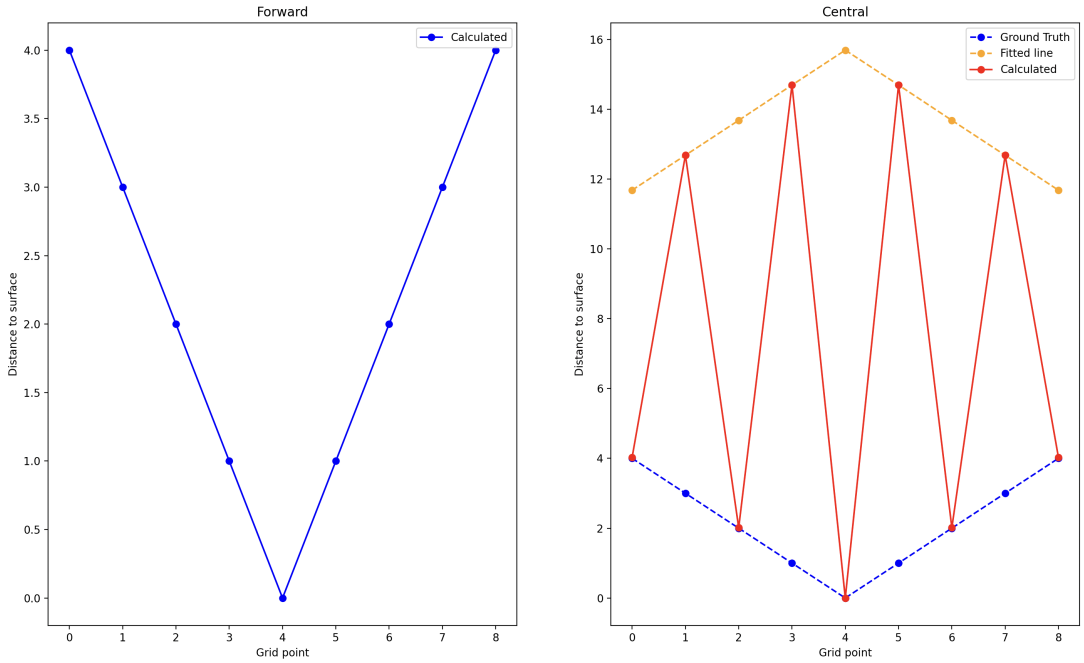


Figure A.3: Comparison of finite difference schemes.

A.2 Accuracy of Hallucinated Points

There exist multiple approaches for generating hallucinated points, each with its advantages and drawbacks. Our current approximation is computationally inexpensive but lacks accuracy [21], particularly as a function of the angle of incidence. As illustrated in Figure A.6a, when the incidence angle deviates from 90°, the discrepancy in approximation becomes evident:

$$d_{\text{true}} = x, \tag{A.2}$$

$$d_{\text{hallucinate}} = \frac{x}{\cos(\alpha)}, \tag{A.3}$$

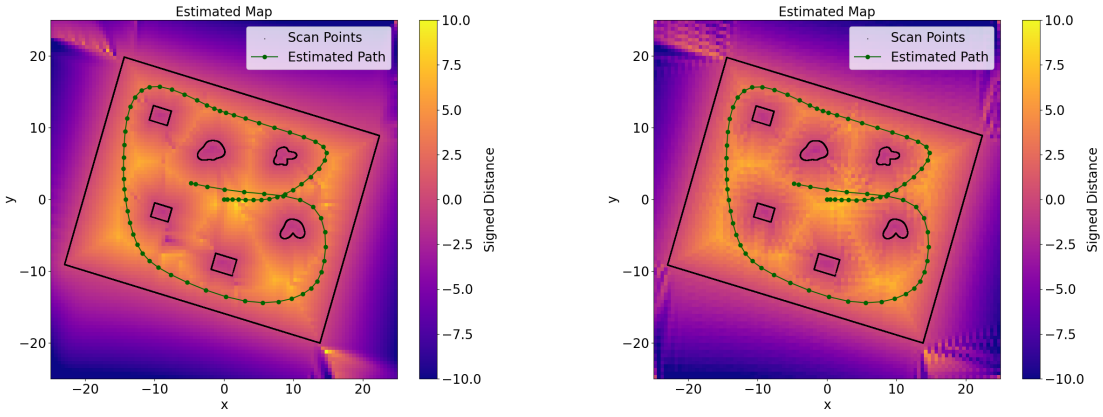
$$\text{err} = |d_{\text{true}} - d_{\text{hallucinate}}|, \tag{A.4}$$

$$= \left| x - \frac{x}{\cos(\alpha)} \right|, \tag{A.5}$$

$$= \left| x \left(1 - \frac{1}{\cos(\alpha)} \right) \right|. \tag{A.6}$$

At an incidence angle of 90°, this error is zero, whereas at 45°, the factor increases to $\sqrt{2}$. At 30°, the factor reaches 2, and it continues to grow for smaller angles. Whether this inaccuracy significantly impacts the system depends on the typical distribution of incidence angles encountered in practice, necessitating an empirical evaluation.

A straightforward improvement involves bounding our estimate by the nearest point in the scan. Since the actual distance must be at most this nearest-neighbor distance, this refinement can mitigate overestimations in the hallucination process.



(a) Forward Difference

(b) Central Difference

Figure A.4: Comparison of maps produced after 25 iterations using two different derivative approximations. The forward difference method produces a smoother map, while the central difference method shows alternating behavior.

Another potential inaccuracy arises from cases illustrated in Figure A.6b, where the hallucinated point is closer to an unobserved surface than the one hit by the laser.

A more refined bounding strategy is shown in Figure A.7. The true distance is constrained above by the nearest scan point and below by the distance to the convex hull formed by the scan points and the scan origin. This follows from the assumption that objects are not small enough to fit between scan lines, like the blue circle in the figure, as we otherwise cannot account for them. The closest we could be to any unseen object would be an object just outside the convex hull, as represented by the black ball.

Denoting the distance to the nearest scan point as a and the distance to the convex hull as b, we can assert the following bounds on the true distance d:

$$\min(a,b) \leq d \leq a. \tag{A.7}$$

Moreover, if $a \leq b$, we can confidently state that $d = a$.

While these improvements enhance distance estimation accuracy, they introduce additional computational complexity, requiring nearest-neighbor searches and convex hull computations. For the purposes of this work, we opt for the simplest approximation; however, future research could explore integrating these enhancements to improve accuracy while maintaining computational feasibility.

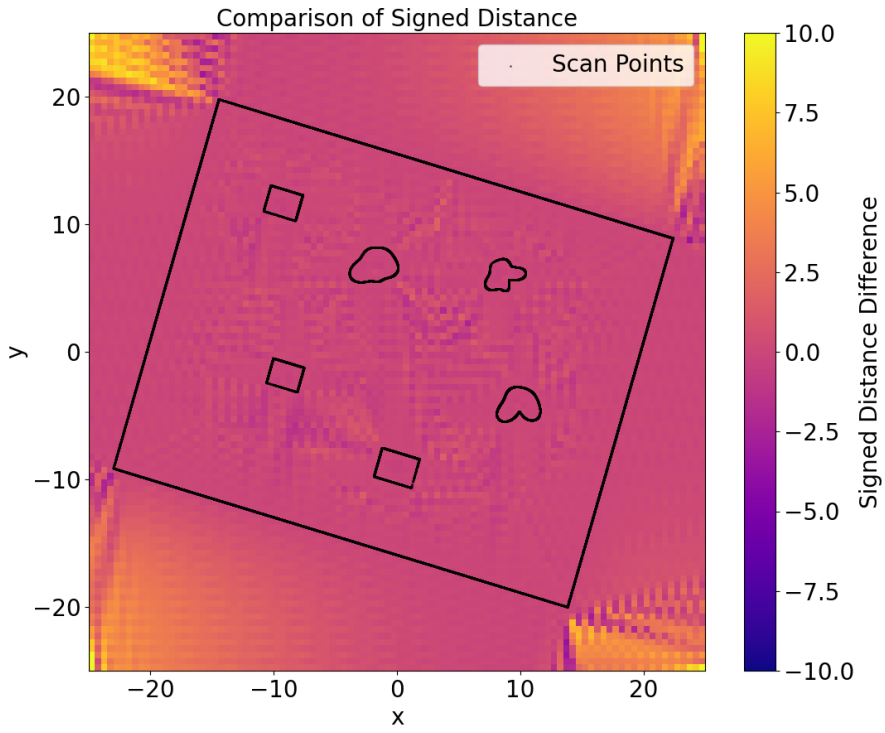
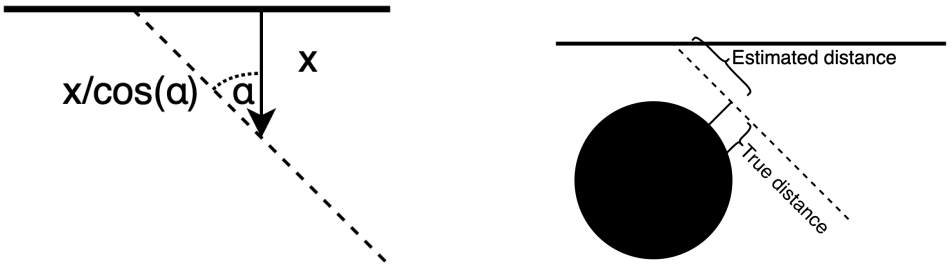


Figure A.5: Chess board pattern in the signed distance error



(a) Approximation in hallucinated points

(b) Error in hallucinated point approximation

Figure A.6: Illustration of hallucinated point errors

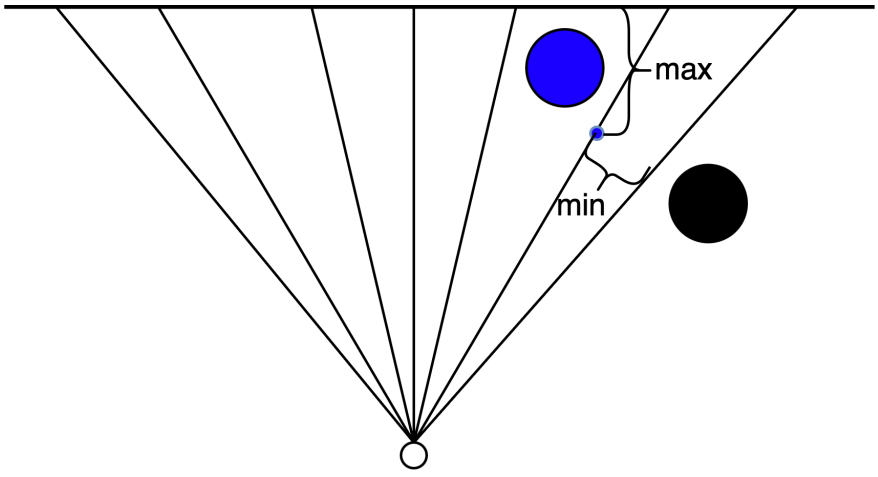


Figure A.7: Refined bounding of hallucinated points

Appendix B

Further Visualizations

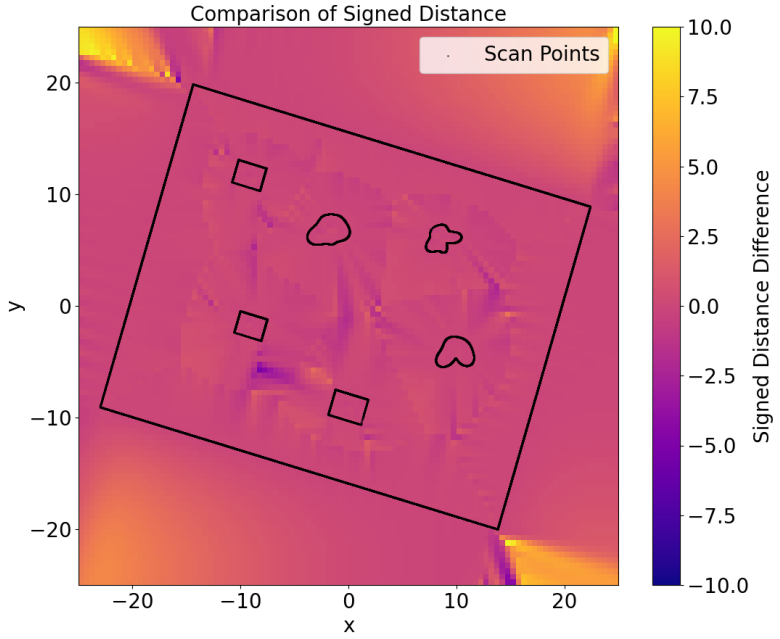


Figure B.1: Visualization of the error of the signed distance value