SoMaSLAM: 2D Graph SLAM for Sparse Range Sensing with Soft Manhattan World Constraints

Jeahn Han[1], Zichao Hu[2], Seonmo Yang[1], Minji Kim[1], and Pyojin Kim[1]

***Abstract*—We propose a novel graph SLAM algorithm for sparse range sensing that incorporates a soft Manhattan world utilizing landmark-landmark constraints. Sparse range sensing is necessary for tiny robots that do not have the luxury of using heavy and expensive sensors. Existing SLAM methods dealing with sparse range sensing lack accuracy and accumulate drift error over time due to limited access to data points. Algorithms that cover this flaw using structural regularities, such as the Manhattan world (MW), have shortcomings when mapping real-world environments that do not coincide with the rules. We propose SoMaSLAM, a 2D graph SLAM designed for tiny drones with sparse range sensing. Our approach effectively maps sparse range data without enforcing *strict* structural regularities and maintains an adaptive graph. We implement the MW assumption as soft constraints, which we refer to as a *soft* Manhattan world. We propose novel soft landmark-landmark constraints to incorporate the soft MW into graph SLAM. Through extensive evaluation, we demonstrate that our proposed SoMaSLAM method improves localization accuracy across diverse datasets and is flexible enough to be used in the real world. We release our source code and dataset on our project page https://SoMaSLAM.github.io/.**

***Index Terms*—SLAM, Range Sensing, Aerial Systems: Perception and Autonomy, Micro/Nano Robots**

I. Introduction

A Range sensor is one of the most effective sensors in simultaneous localization and mapping (SLAM) [1], [2] for accurately recognizing the surrounding environments. Although heavy, bulky, and expensive range sensors, such as Velodyne VLP-16, are very accurate and reliable, they are sometimes not suitable for specific situations. Palm-sized tiny robots like the Crazyflie 2.1 [3] cannot utilize these accurate range sensors due to their limited payload of only 15 g. It is logical to sacrifice accuracy and the number of range sensing points for a lightweight range sensor, leading to the sparse range sensing problem.

Sparse range sensing severely limits the range data, meaning existing SLAM methods not tailored to the problem cannot work properly due to the lack of information. Several prior works [4], [5] tackle this challenge to overcome limited range sensing capacity within particle filter and pose graph SLAM frameworks. Despite these efforts, drift error accumulates over time, and the methods mentioned do not take advantage of repeated structural regularities.

Manuscript received: February 28, 2025; Revised: May 16, 2025; Accepted: June 12, 2025. This letter was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments. The work of Pyojin Kim was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00358374). *(Corresponding author: Pyojin Kim.)*

[1]Department of Mechanical and Robotics Engineering, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, South Korea. {jeahnhaan,seonmo.yang,minji0110}@gm.gist.ac.kr pjinkim@gist.ac.kr

[2]Department of Computer Science, University of Texas at Austin, Texas 78712, USA. zichao@utexas.edu

Digital Object Identifier (DOI): see top of this page.

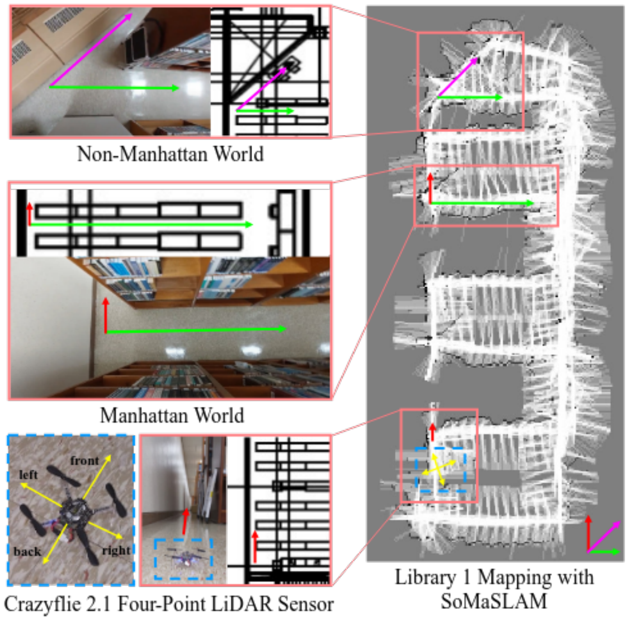

Non-Manhattan World
Manhattan World
front
left
back
right
Crazyflie 2.1 Four-Point LiDAR Sensor
Library 1 Mapping with SoMaSLAM

Fig. 1. Our SoMaSLAM results (right) where non-Manhattan (top-left) and Manhattan (left-middle) worlds co-exist in a structured environment. We obtain sparse data by a low-cost, lightweight (2.3 g), four radially-spaced ToF range sensor attached to the Crazyflie [3] nano drone (bottom-left).

Some SLAM methods [6], [7], [8] integrate structural regularities such as Manhattan world (MW) [9], where all planes must be parallel or orthogonal to each other. These algorithms neutralize the drift error by effectively utilizing indoor structural patterns and consistently produce accurate results, but fail when pre-assumed structural models are broken. An open door or an out-of-place desk can be enough to disrupt the regulated conditions of a Manhattan world. Other structural models like Atlanta world [10] or a mixture of Manhattan frames [11] allow a bit more flexibility. Nonetheless, the fact that all features must obey a mandatory rule when using structural regularities makes SLAM algorithms vulnerable to failure when using real-world datasets.

To address these issues, we propose SoMaSLAM, a novel 2D graph SLAM for sparse range sensing leveraging structural regularities, which extends the previous 2D graph SLAM [5] to achieve more accurate and consistent performance in structured environments. In particular, we formulate the objective function in pose graph optimization (PGO) that *encourages*, rather than forces, nearby walls to be orthogonal or parallel to each other. We propose new soft constraints [12] between landmarks called the *soft* Manhattan world (MW) [13], devoid of *hard* constraints [14], [8]. Our soft MW approach in graph SLAM allows more flexibility in structured environments and molds the map into a version that best satisfies all conditions, as shown in Fig. 1. The proposed soft landmark-landmark constraints are easily integrated into any pose-graph SLAM system with line or plane landmarks, enhancing robustness and accuracy in non-Manhattan environments.

Our main contributions are as follows:

- We leverage structural regularities in man-made environments for a sparse range sensing problem in the pose graph SLAM framework.
- We propose a new soft landmark-landmark constraint on the PGO-based soft MW, allowing more flexibility in structured environments.
- We evaluate SoMaSLAM on both author-collected and public datasets [15], comparing it with state-of-the-art SLAM methods. We also provide mathematical proof of the superiority of soft MW over traditional MW in graph SLAM and release our datasets and code publicly.

To the best of our knowledge, this is the first 2D graph SLAM with soft constraints between landmarks inspired by the soft MW assumption.

II. Related Work

Graph SLAM, which frames SLAM as a graph, is a popular approach for its nonlinear optimization and flexible graph structure. We dive deeper into SLAM with sparse range sensing, and SLAM with structural regularities such as the Manhattan world assumption [9].

SLAM with sparse sensing is a category of SLAM where the robot receives and uses very few data points from sensors, often resulting in challenges related to localization and mapping accuracy. Despite the significance of SLAM with sparse sensing, relatively few works have been published in this area. Beevers et al. [4] addresses the SLAM problem with sparse sensing, utilizing a Rao-Blackwellized particle filter [16] to combine data from several scans to increase density. Yap et al. [17] tackle SLAM using low-cost, noisy sonar sensors by employing particle filters and a line-segment-based map with an orthogonality assumption. Zhou et al. [5] address SLAM with sparse sensing by utilizing a graph SLAM with a novel front-end to replace scan matching and improved loop closing in the back-end, specifically designed for sparse and uncertain range data. In NanoSLAM [18], new high-density ToF sensor hardware is introduced for Crazyflie nano drones [3] to overcome sparse sensing. Despite these approaches, SLAM with sparse sensing frequently suffers from a lack of accuracy due to the inherent limitation of relying on minimal data points.

Some odometry and SLAM methods incorporate structural regularities to enhance performance, but at the expense of mapping versatility in diverse environments. The Manhattan world [9] is a fundamental structural regularity and a widely favored choice for integration into SLAM algorithms. Liu et al. [14] present a stereo visual SLAM for man-made environments, utilizing a two-stage Manhattan frame (MF) tracking with line features for absolute orientation estimation based on ORB-SLAM2 [19]. Jeong et al. [8] outline a SLAM algorithm that supplements the limitations of RGB-D cameras using a four-point LiDAR, helping to build a reliable global MW map and estimate 6-DoF camera poses. A step up from the Manhattan world is the Atlanta world (AW) [10]. In an Atlanta world, all walls remain orthogonal to the ground plane, but no longer need to maintain a fixed relationship (orthogonality) with one another. Joo et al. [7] propose a linear RGB-D SLAM designed for the Manhattan and Atlanta world by accommodating a vertical axis and multiple horizontal directions with planar features in the Kalman filter framework. Yet, these SLAM methods that utilize structural regularities, such as MW and AW, lack flexibility in representing the surrounding structured environments. The environment must suit the rules perfectly for the methods to perform well. They are prone to failure when mapping non-structurally regulated environments. Moreover, many of the discussed methods rely on an RGB-D camera, which is often unavailable for nano drones [18] like Crazyflie.

III. Proposed Method

Our proposed method, SoMaSLAM, builds on Efficient Graph SLAM (EG-SLAM) [5]. The shortage of data negatively impacts the results. We incorporate a soft MW assumption through landmark-landmark constraints. Sec. III-A provides a short explanation of [5]. Sec. III-B explains our proposed method, which is also shown in Fig. 2 as a flowchart.

A. Efficient 2D Graph SLAM for Sparse Sensing

We briefly summarize EG-SLAM and recommend referring to [5] for a detailed explanation. EG-SLAM builds upon a graph-based framework with key improvements both in its front-end and back-end. In the front-end, the method accumulates sparse range data over multiple poses to form a multiscan [4], later turned into line segments [20], [21]. Each line is parameterized as polar coordinates $l = (\rho, \theta)$ and is associated with an existing landmark in the graph. If there is no appropriate landmark, the line segment creates a new landmark. This step accounts for lines 1 and 2 in Algorithm 1. A pose and a landmark form a pose-landmark constraint, which is then inserted into the landmark graph. This step accounts for line 3 in Algorithm 1. When the pose-landmark constraint is formed, the error term for the constraint is calculated as follows [5].

$$e_l(\mathbf{x}_i, \mathbf{l}_j) = \mathbf{v}_{ij} - f(\mathbf{x}_i^{-1}, \mathbf{l}_j) \tag{1}$$

where $\mathbf{v}_{ij} = (\rho_{ij}, \alpha_{ij})$ is the measurement of the landmark $\mathbf{l}_j$ in the frame of $\mathbf{x}_i$, parameterized in polar form by its perpendicular distance ρ_{ij} and normal angle α_{ij}. $f(\mathbf{x}_i^{-1}, \mathbf{l}_j)$ transforms the current estimate of landmark $\mathbf{l}_j$ from the global frame to the frame of $\mathbf{x}_i$. The key idea from [5] is to maintain two graphs, a landmark graph and a pose-only graph. The landmark graph is comprised of poses and landmarks. It is responsible for local consistency by creating and optimizing

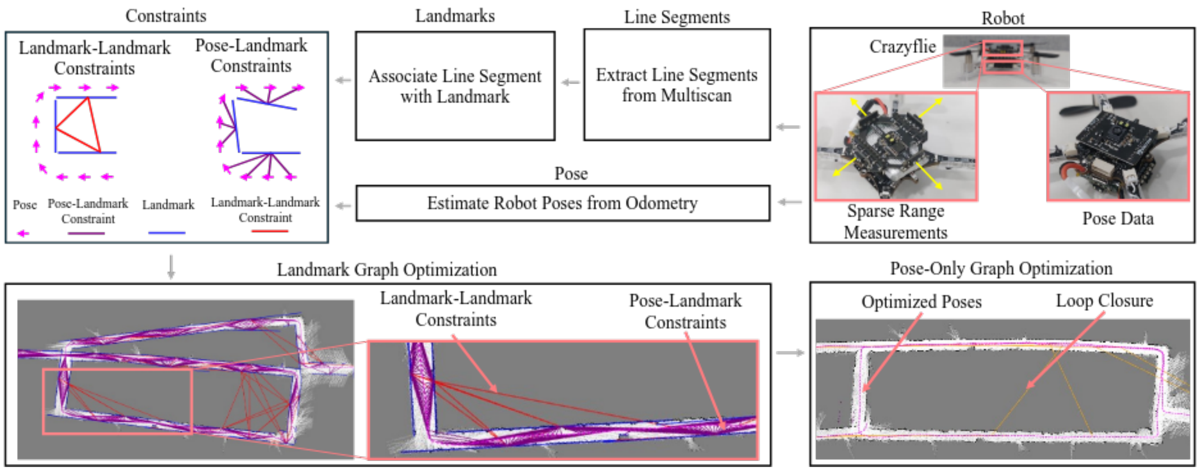

Constraints
Landmark-Landmark Constraints
Pose-Landmark Constraints
Pose
Pose-Landmark Constraint
Landmark
Landmark-Landmark Constraint
Landmarks
Associate Line Segment with Landmark
Line Segments
Extract Line Segments from Multiscan
Robot
Crazyflie
Sparse Range Measurements
Pose Data
Pose
Estimate Robot Poses from Odometry
Landmark Graph Optimization
Landmark-Landmark Constraints
Pose-Landmark Constraints
Pose-Only Graph Optimization
Optimized Poses
Loop Closure

Fig. 2. Overview of the proposed SoMaSLAM. We construct poses (magenta arrows) and landmarks (blue) in graph SLAM from odometry and line segments given sparse range measurements. We generate and impose pose-landmark constraints (purple edges) and soft landmark-landmark constraints (red edges) between poses as well as between landmarks. Loop closing (light brown edges) occurs during pose-only graph optimization.

Algorithm 1 Creating Constraints in Landmark Graph

```
1: for each segment in segments do
2:     Associate line segments with landmark
3:     Create a pose-landmark constraint and insert into landmark graph
4:     for landmark in landmarks do
5:         if requirements are satisfied then
6:             Create a landmark-landmark constraint and insert into landmark graph
7:         end if
8:     end for
9: end for
```

constraints. The pose-only graph deals with only poses and does not deal with landmarks. The goal of the pose-only graph is to use the updated poses from the landmark graph to perform loop closing. The specifics of this unique dual graph procedure are mentioned in [5]. We work only with the landmark graph and leave the pose-only graph unchanged.

B. Landmark-Landmark Constraints on Soft MW

1) Landmark-Landmark Constraints: Measurements between poses, as well as between poses and landmarks, are essential for forming constraints in graph SLAM. Landmark-landmark constraints cannot be formulated in the same way as pose-landmark and pose-pose constraints, because they lack a direct reference to a moving pose.

2) Soft Manhattan World Assumption: Many indoor environments resemble a Manhattan world, where walls are either parallel or orthogonal to each other. However, most indoor environments partially follow this structure. In particular, diagonal walls deviate from the ideal Manhattan world model.

3) Iterating over Existing Landmarks (line 4): We go over existing landmarks to find a landmark capable of forming constraints with the landmark associated with the line segment in question. The landmark associated with the line segment is not always the most recently created landmark.

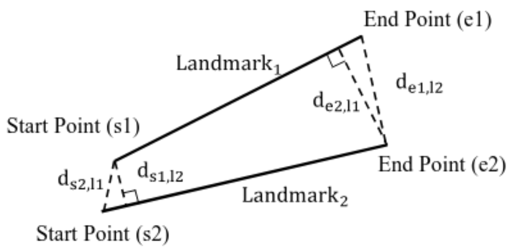


Fig. 3. Illustration of the four cases for computing the shortest distance between two line segments. Each case measures the smallest distance from a point (start or end point of a landmark) to a line segment of the other landmark. The shortest distance between the two line segments is the minimum of these four values.

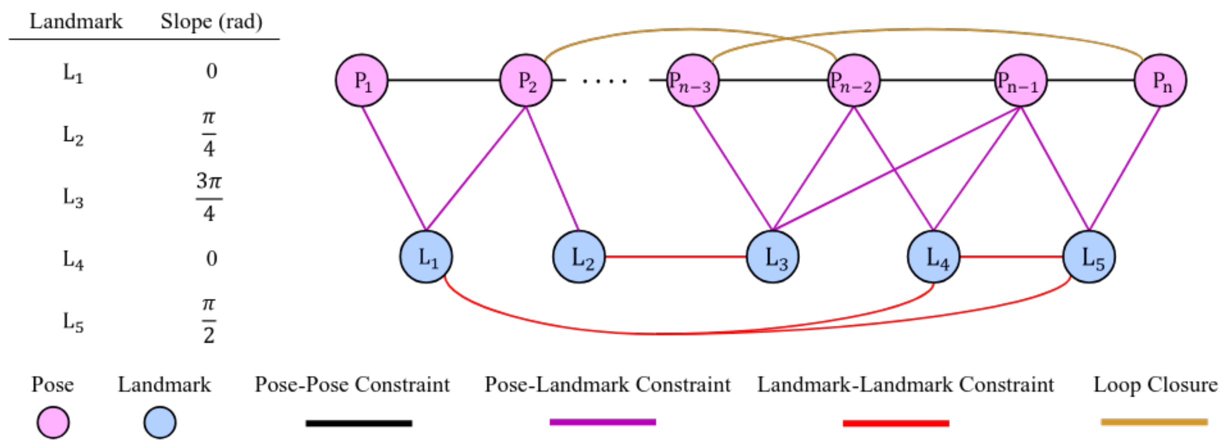
4) Requirements for Creating Landmark-Landmark Constraints (line 5): This process determines whether the two landmarks are suitable for a landmark-landmark constraint. We use three criteria to decide:

1) Two landmarks are near each other
2) Both landmarks are *significant* landmarks
3) The landmark-landmark constraint is not oversaturated

Criterion 1 establishes local landmark-landmark constraints from both the spatial and temporal perspectives. Two landmarks are considered spatially local if the two line segments intersect or if the shortest distance between the two line segments is below a given threshold. If two line segments do not intersect, the minimum distance between the line segments involves an endpoint [22]. We simultaneously consider all four cases $d_{s1,l2}, d_{e1,l2}, d_{s2,l1}, d_{e2,l1}$ shown in Fig. 3 where $d_{i,j}$ is the shortest distance between point i and line segment j. The shortest distance between two line segments is calculated as $\min(d_{s1,l2}, d_{e1,l2}, d_{s2,l1}, d_{e2,l1})$.

Temporally, a landmark is considered local if its ID value—assigned in increasing order based on creation time—is close to that of another landmark. Both landmarks must be temporally and spatially local for Criterion 1 to be satisfied.

Criterion 2 focuses on selecting major landmarks that effectively represent the environment. To qualify, a landmark must exceed a length threshold and have a sufficient number of



Landmark	Slope (rad)
L_1	0
L_2	$\frac{\pi}{4}$
L_3	$\frac{3\pi}{4}$
L_4	0
L_5	$\frac{\pi}{2}$

Fig. 4. Illustration of a landmark graph for SoMaSLAM. Pose-landmark constraints (purple edges), soft landmark-landmark constraints (red edges), and loop closure constraints (light brown edges) make up the constraints for the graph. The orientation of the landmarks (blue) and the constraints between them are shown as examples above each landmark node.

pose-landmark constraints connected to it. Shorter landmarks and those with fewer pose-landmark constraints are often misrepresentations of walls.

Criterion 3 addresses the landmark-landmark constraint oversaturation by tracking and preventing an excessive number of repeated constraints between the same pair of landmarks. Not setting a limit on the number of constraints has resulted in significant performance deterioration.

5) Creating a Landmark-Landmark Constraint (line 6):

$$\left|\theta_{\mathbf{2}} - \theta_{\mathbf{1}} - \frac{k}{2}\pi\right| < \epsilon \Rightarrow \boldsymbol{\Delta}\theta_{\mathbf{ideal}} = \frac{k}{2}\pi \tag{2}$$

We introduce artificial ideal values to use as measurements to compensate for the absence of actual measured data. We focus on the orientation of the landmarks. The relationship between landmarks $\mathbf{l}_1$ and $\mathbf{l}_2$ decides $\boldsymbol{\Delta}\theta_{\mathbf{ideal}}$ in Eq. (2), where $k \in \{-2, -1, 0, 1, 2\}$, and ϵ is a threshold. $\boldsymbol{\Delta}\theta_{\mathbf{ideal}}$ is the difference in orientation in an ideal MW where landmarks that are *almost* parallel or orthogonal are treated as *exactly* parallel or orthogonal.

We draw inspiration from pose-landmark constraints to define the error term for landmark-landmark constraints. In pose-landmark constraints, the error term captures the difference between the measured and estimated landmark positions relative to the pose frame. In our approach to creating landmark-landmark constraints, we treat $\mathbf{l}_1$ as the pose and calculate the necessary values for $\mathbf{l}_2$ as if forming a pose-landmark constraint. The measured and estimated orientation of $\mathbf{l}_2$ is relative to $\mathbf{l}_1$. The ideal orientation of $\mathbf{l}_2$ ($\theta_{\mathbf{2,ideal}}$) is computed as $\theta_{\mathbf{2,ideal}} = \boldsymbol{\Delta}\theta_{\mathbf{ideal}} + \theta_{\mathbf{1}}$ and is treated as the measurement.

$$e_{ll}(\mathbf{l}_1, \mathbf{l}_2) = \theta_{\mathbf{2,ideal}} - \theta_{\mathbf{2}} \tag{3}$$

$$e_{ll}(\mathbf{l}_1, \mathbf{l}_2) = \theta_{\mathbf{1}} - \theta_{\mathbf{2}} - \theta_{ideal} \tag{4}$$

We define the error term for landmark-landmark constraints as Eq. (3) where e_{ll} is the error term, $\theta_{\mathbf{2,ideal}}$ serves as the measured value and $\theta_{\mathbf{2}}$ is the estimated value. $\theta_{\mathbf{2,ideal}}$ is a constant and the only variable inside Eq. (3) is $\theta_{\mathbf{2}}$. We use only one variable in the constraint error term rather than using two variables like Eq. (4) where $\theta_{\mathbf{1}}, \theta_{\mathbf{2}}$ are both variables.

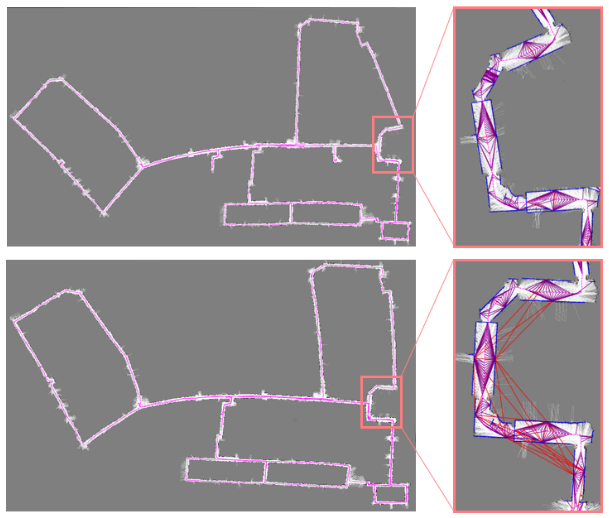


Fig. 5. SoMaSLAM results on MIT Killian (11 points) with two variables in the constraint error term e_{ll} (top) and one variable (bottom) in the constraint error term. Formulating a landmark-landmark constraint with a single model parameter shows better performance.

Setting our error term with only one variable, shown in Eq. (3), produces a smaller overall error compared to having two variables inside the error term, such as Eq. (4). Increasing the degrees of freedom and complexity of the optimization problem can lead to convergence difficulties and suboptimal solutions. This is shown in Fig. 5.

$$\mathbf{\Omega_{i,j}} = \begin{bmatrix} \mathbf{len}_1 + \mathbf{len}_2 & 0 \\ 0 & \mathbf{len}_1 + \mathbf{len}_2 \end{bmatrix} \tag{5}$$

In graph SLAM, an edge includes both the error term and the information matrix, the latter reflecting the significance of the constraint. We define our matrix to assign the appropriate importance to each constraint. Eq. (5) represents the information matrix for a landmark-landmark constraint between landmarks $\mathbf{l}_i$ and $\mathbf{l}_j$, with $\mathbf{len}_i$ denoting the length of landmark $\mathbf{l}i$. Shorter landmarks yield smaller values in the information matrix, while longer landmarks, often representing key features like walls, have larger values on the diagonal. This ensures the optimization process appropriately prioritizes necessary constraints. Fig. 4 shows the pose graph representation for our proposed SoMaSLAM. It is key to notice that only suitable pairs of landmarks have formed landmark-landmark constraints.

Multiple landmark-landmark constraints per pair are maintained to preserve temporal measurement variations, preventing information loss from aggregation that can impair optimization. This results in a constraint count significantly exceeding the number of landmarks. Quantitative analyses and exact counts are provided in the supplementary material.

6) Graph Optimization: We use $e_{ll}(\mathbf{l}_i, \mathbf{l}_j)$ and $\mathbf{\Omega_{i,j}}$, defined previously, to formulate the objective function as follows:

$$G(X) = \sum_{ij} e_{ll}(\mathbf{l}_i, \mathbf{l}_j)^T \mathbf{\Omega_{i,j}} e_{ll}(\mathbf{l}_i, \mathbf{l}_j) + F(X) \tag{6}$$

$$F(X) = \sum_{i} \|e_o(\mathbf{x}_i, \mathbf{x}_{i+1})\|^2_{\Sigma_i} + \sum_{ij} \|e_l(\mathbf{x}_i, \mathbf{l}_j)\|^2_{\Sigma_{ij}} \tag{7}$$

where *F(X)* is the summation of the squared odometry and landmark errors, weighted by their corresponding covariances as shown in Eq. (7). The exact definition is mentioned thoroughly in EG-SLAM [5]. The Levenberg-Marquardt solver in g2o [23] is used for pose graph optimization.

7) Computational Efficiency of Soft Manhattan World Constraints over Strict Manhattan World Constraints in Graph SLAM: Our method introduces the soft Manhattan world (SoMaSLAM), which outperforms methods without structural regularities, such as EG SLAM. SoMaSLAM also accommodates non-Manhattan environments, unlike FL SLAM. The soft Manhattan world provides mathematical advantages over the strict Manhattan world, as demonstrated from both optimization and factor graph perspectives.

Graph SLAM typically formulates a non-linear function through pose-pose and pose-landmark constraints, optimized using Gauss-Newton or Levenberg-Marquardt. However, enforcing strict Manhattan constraints introduces an additional restriction: landmark orientation must be one of four discrete values. This disrupts the continuous optimization framework, significantly increasing computational complexity.

Incorporating a strict Manhattan world assumption into graph SLAM converts the original nonlinear programming (NLP) problem into a mixed-integer nonlinear programming (MINLP) problem formulation, known to be NP-hard [24]. From a factor graph perspective, this introduces discrete variables by constraining landmark orientations to specific discrete values, creating a hybrid discrete-continuous optimization problem. Consequently, specialized solvers such as Sequential Quadratic Programming (SQP) are required, increasing computational complexity [25]. In contrast, SoMaSLAM adopts a soft Manhattan world assumption, maintaining a fully continuous NLP formulation compatible with standard nonlinear optimization methods. This simplification significantly reduces computational overhead and improves scalability. Experiments comparing SoMaSLAM to a strict Manhattan world approach on synthetic datasets confirmed these benefits, showing lower runtime and memory usage as landmark counts increased. Detailed results are presented in the supplementary material.

TABLE I
EVALUATION RESULTS ON RADISH DATASET (4 PTS)[1]

	SoMaSLAM	**EG-SLAM [5]**	**FL-SLAM [8]**
Aces			
Abs. Trans. (m)	0.04 ± 0.05	**0.04 ± 0.05**	0.55 ± 2.68
Abs. Rot. (°)	**1.04 ± 1.34**	1.04 ± 1.36	2.33 ± 6.51
Intel Lab			
Abs. Trans. (m)	**0.13 ± 0.21**	2.80 ± 8.98	7.37 ± 14.35
Abs. Rot. (°)	**2.71 ± 3.01**	15.19 ± 36.00	39.17 ± 53.80
MIT Killian			
Abs. Trans. (m)	**0.91 ± 4.49**	17.84 ± 46.25	50.32 ± 128.15
Abs. Rot. (°)	**2.19 ± 3.77**	15.81 ± 37.69	9.26 ± 27.03
MIT CSAIL			
Abs. Trans. (m)	**0.39 ± 1.24**	2.53 ± 9.00	1.68 ± 5.29
Abs. Rot. (°)	**4.10 ± 5.68**	11.45 ± 28.23	5.07 ± 8.24
Freiburg-079			
Abs. Trans. (m)	**0.28 ± 0.72**	0.52 ± 1.18	6.50 ± 13.67
Abs. Rot. (°)	**3.86 ± 6.23**	4.77 ± 6.16	30.29 ± 45.53

IV. EXPERIMENTS

To evaluate the effectiveness of the proposed SoMaSLAM, we conduct extensive testing using both the Radish dataset [26] and author-collected datasets. We compare its performance against EG-SLAM [5] and FL-SLAM [8].

A. Running LiDAR Datasets on FL-SLAM

FL-SLAM is a SLAM method designed for mapping Manhattan worlds using both LiDAR and RGB-D data. It extends the approach of [7] by addressing the unreliability of RGB-D depth data in open spaces through precise LiDAR measurements. In non-open environments, FL-SLAM integrates both RGB-D and LiDAR for pose estimation while enforcing a strict Manhattan-world assumption. In open spaces, it relies solely on LiDAR to maintain this assumption. Since the Radish datasets and our collected datasets contain only LiDAR measurements, we classify all instances in our dataset as open spaces. This ensures compatibility with FL-SLAM by effectively disabling the use of RGB-D data.

B. Radish Datasets

Fig. 6 and Table I show the qualitative and quantitative results for running Radish datasets on SoMaSLAM, EG-SLAM, and FL-SLAM. We sample 4 uniformly spaced measurement values out of the available values to simulate a sparse sensor dataset. We refer to using only 4 points from the dataset as 4pts. Numerical analysis was conducted using the method and ground truth values provided by [15]. FL-SLAM is unsuccessful at mapping Radish datasets. This is due to FL-SLAM being highly dependent on accurate pose values acquired from the ARKit. The absolute translational error and absolute rotational error in Table I are often one to two orders of magnitude

[1]While the rounded values for absolute translation (m) in the Aces row appear identical for SoMaSLAM and EG-SLAM, the full-precision values reveal that EG-SLAM has a slightly lower error, indicating better performance

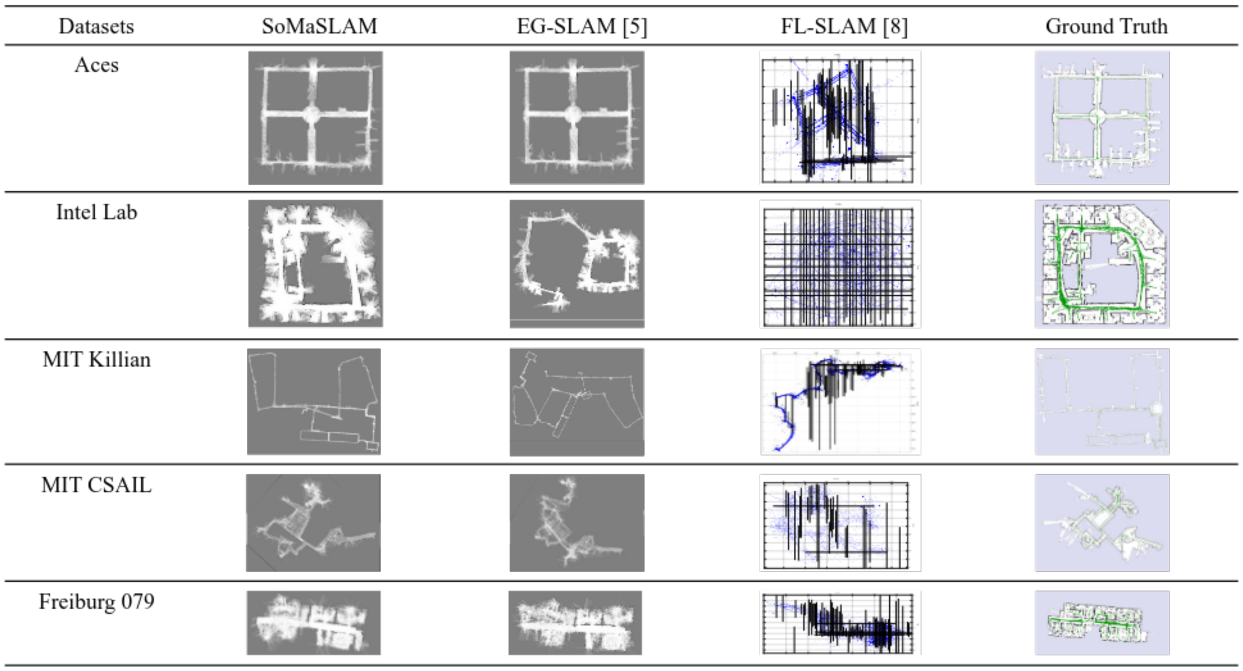


Fig. 6. Each column denotes SLAM results on Radish datasets (4pts) with SoMaSLAM, EG-SLAM [5], FL-SLAM [8], and the ground truth for qualitative comparison. FL-SLAM fails since the strict MW assumption is broken. EG-SLAM is inaccurate due to the four range measurements (4pts) being extremely sparse, whereas the proposed SoMaSLAM is comparable to the ground truth by utilizing the soft MW constraints, even with only 4pts.

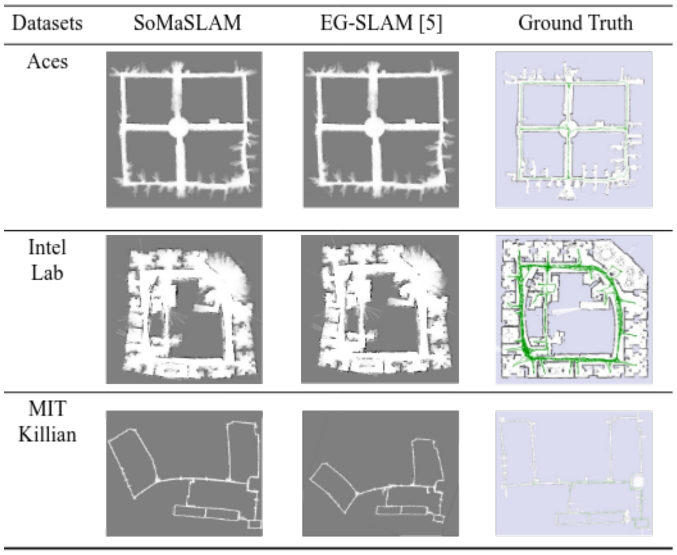


Fig. 7. SLAM results on Radish datasets (11pts) with SoMaSLAM and EG-SLAM [5]. The ground truth is on the right for qualitative comparison. The qualitative difference between SoMaSLAM and EG-SLAM [5] is negligible.

larger than EG-SLAM and SoMaSLAM across all Radish datasets. EG-SLAM has lower accuracy than SoMaSLAM across most datasets, except for Aces, a relatively short dataset. Fig. 7 and Table II show results for running Radish datasets on SomaSLAM and EG-SLAM. While the 4-point dataset simulates sparse sensing suitable for nano drones, the 11-point dataset provides a denser subset, consistent with the 11-point comparison approach used in [5] for benchmarking SoMaSLAM. Soft MW constraints in SoMaSLAM effectively reduce error accumulation. Fig. 8 compares SoMaSLAM and EG-SLAM trajectories on the MIT Killian dataset, showing SoMaSLAM closer to the ground truth. These constraints also improve loop closing, as seen in Figs. 9 and 10. Drift error in EG-SLAM limits loop closing, while SoMaSLAM mitigates drift with landmark-landmark constraints. Results demonstrate improved loop closure success and reduced trajectory drift on nano drone datasets with sparse sensing.

TABLE II
EVALUATION RESULTS ON RADISH DATASET (11 PTS)

	SoMaSLAM	**EG-SLAM [5]**
Aces		
Abs. Trans. (m)	$\mathbf{0.04 \pm 0.05}$	0.05 ± 0.05
Abs. Rot. (°)	$\mathbf{1.15 \pm 1.43}$	1.16 ± 1.44
Intel Lab		
Abs. Trans. (m)	$\mathbf{0.09 \pm 0.13}$	0.09 ± 0.13
Abs. Rot. (°)	2.41 ± 2.50	$\mathbf{2.38 \pm 2.54}$
MIT Killian		
Abs. Trans. (m)	$\mathbf{0.07 \pm 0.15}$	0.93 ± 3.39
Abs. Rot. (°)	$\mathbf{1.98 \pm 3.50}$	2.29 ± 4.25

TABLE III
TYPES OF PROCESSING TIME PER FRAME IN MS

	Aces	**Intel Lab**	**MIT Killian**
EG-SLAM [5]			
Average	0.8179	2.6253	4.7549
Max. Frontend	20.454	17.241	98.143
Max. Backend	73.823	276.36	85.262
SoMaSLAM			
Average	1.0668	2.7778	5.5285
Max. Frontend	20.454	15.181	63.079
Max. Backend	113.42	275.69	369.67

We compare SoMaSLAM to dense B-spline SLAM [27] on the Radish dataset [26]. Although B-spline SLAM achieves lower errors, SoMaSLAM attains qualitatively comparable results under extreme sparsity, utilizing 98.2% fewer data points. Refer to the supplementary material for details.

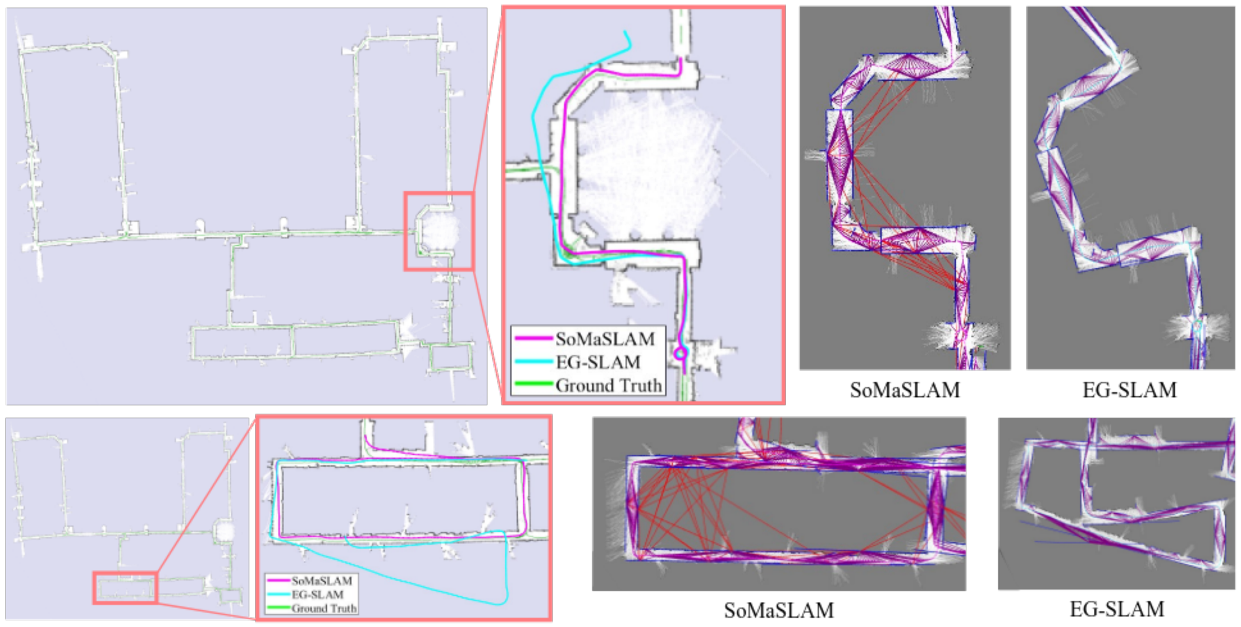


Fig. 8. Pose comparison of SoMaSLAM (magenta), EG-SLAM (cyan), and the ground truth (green) on the MIT Killian dataset (4pts). Landmark graphs of SoMaSLAM (dotted magenta) and EG-SLAM (dotted cyan) are on the right. The landmark-landmark constraints (red edges) enhance mapping accuracy, shown by the near-perfect pose trajectory from SoMaSLAM.

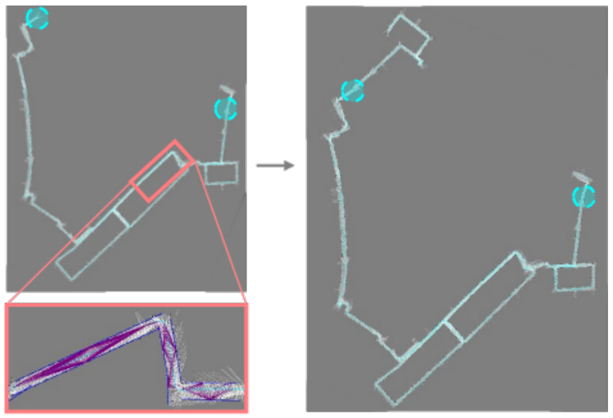


Fig. 9. MIT Killian (4 points) partially mapped with EG-SLAM. Loop closure opportunities are translucent circles, and successful loop closures are solid circles. EG-SLAM fails to perform successful loop closing, shown by the translucent circles on the right, due to excessive error accumulation.

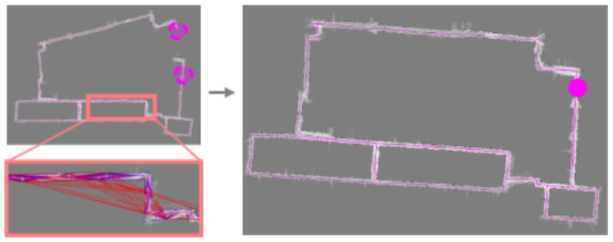


Fig. 10. MIT Killian (4 points) partially mapped with SoMaSLAM. Loop closure opportunities are translucent circles, and successful loop closures are solid circles. SoMaSLAM performs successful loop closing, shown by the solid circle on the right, due to reduced error accumulation from soft landmark-landmark constraints.

Our experiments use VL53L1x ToF sensors on nano drones and simulate denser 8x8 depth sensor (ST VL53L8CX) configurations for potential accuracy improvements. Details on the implementation and results are in the supplementary materials.

Table III compares the mean processing time per frame for EG-SLAM and SoMaSLAM on an Intel Core i5-12400F PC. A frame is a single cycle of sensor data processed in the SLAM system. The average time difference is negligible. The frontend processes sparse sensor data to estimate local pose relations, and the backend optimizes the global pose graph with loop closures. SoMaSLAM has a higher maximum backend processing time for larger datasets like Intel Lab and MIT Killian, while EG-SLAM shows a significantly higher maximum frontend processing time—up to 55.6% for MIT Killian and 13.6% for Intel Lab compared to SoMaSLAM.

Onboard backend optimization is infeasible with current hardware due to high computational and memory demands. While real-time constraints are not met, sparse and hierarchical optimization methods show promise for future onboard implementation. See supplementary materials for details.

We perform a hyperparameter study on seven key factors influencing landmark constraints: spatial and temporal thresholds, landmark length, constraint limits, angular thresholds, and error formulation complexity. Setting an effective spatial threshold (15 m) reduces error by around 40%, and a temporal gap (10 landmark IDs) decreases error by around 30% compared to reasonable but suboptimal baseline values. A landmark length threshold (0.3 m) lowers rotational errors by around 210%, while limiting landmark constraints per pair improves translational accuracy by up to 45%. Optimal angle thresholds (10°) balance accuracy and flexibility effectively. Single-variable constraint formulations enhance convergence and reduce translational error by approximately 460%. Specific numerical values and additional details are provided in the supplementary material.

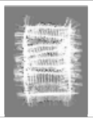
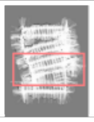
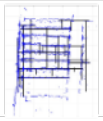
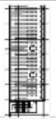
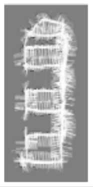
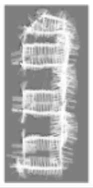
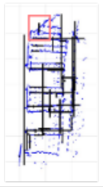
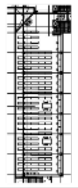
Datasets	SoMaSLAM	EG-SLAM [5]	FL-SLAM [8]	Ground Truth
Library 1				
Library 2				

Fig. 11. Library 1 and Library 2 dataset (4pts) results with SoMaSLAM, EG-SLAM [5], and FL-SLAM [8]. FL-SLAM plots the point cloud (blue) and creates walls (black). EG-SLAM struggles to map Library 1, and FL-SLAM fails to map the diagonal walls present in Library 2.

C. Author-Collected Datasets

We use the Crazyflie nano drone [3] to collect sparse range data from the 2nd floor of a library at our institution. We attach an optical flow sensor, a flow deck (1.6 g) for odometry data, and a multi-ranger deck (2.3 g) to collect sparse range measurements from the front, back, left, and right of the Crazyflie. Since they are low-cost, lightweight, and compact-sized sensors, the range data is highly sparse and insufficiently accurate. We gather two notable datasets, Library 1 and Library 2, to compare with EG-SLAM and FL-SLAM. The four-point range data from the Crazyflie provides a challenging scenario, serving as a robust evaluation to see if the algorithm is viable with real-world sparse data. EG-SLAM fails to accurately map in mapping a structured world when dealing with Library 1, shown in Fig. 11. However, SoMaSLAM utilizes the distinct Manhattan world-like environment to map the environment accurately. FL-SLAM also creates suitable walls that align with the Manhattan world. For Library 2, SoMaSLAM and EG-SLAM have nearly identical results. FL-SLAM partially succeeds at mapping Manhattan world walls. However, it fails to detect non-MW features at the top left and instead creates walls aligning with the initially declared Manhattan world. In other words, FL-SLAM does not attempt to map the diagonal wall at the top left of Library 2. FL-SLAM is made for a Manhattan world and cannot accurately map a non-Manhattan world, as shown through Library 2.

We also validate SoMaSLAM's robustness in complex indoor scenes with curved and irregular walls, further demonstrating effective mapping beyond strict Manhattan assumptions. Please refer to the supplementary materials for more information.

V. Conclusion

We introduce two novel ideas in graph SLAM: a soft landmark-landmark constraint and a soft Manhattan world (MW) assumption. We show how our new method improves the accuracy of graph SLAM with sparse sensing and provides more versatility compared to using strict structural regularities. In future work, we aim to integrate multiple drones with soft MW constraints for distributed mapping, enhancing SLAM accuracy.

Acknowledgements

The authors thank the library staff, Yeon Sook Ryu and Yong Kwan Kim, for kindly allowing us to collect data using a Crazyflie drone in the library. We also thank the author of [8] for providing the code for FL-SLAM.

References

[1] J. Zhang, S. Singh, *et al.*, “Loam: Lidar odometry and mapping in real-time.” in *RSS*, 2014.

[2] T. Shan, “Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in *IEEE IROS*, 2018.

[3] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, “Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering,” in *IEEE MMAR*, 2017.

[4] K. R. Beevers and W. H. Huang, “Slam with sparse sensing,” in *IEEE ICRA*, 2006.

[5] H. Zhou, Z. Hu, S. Liu, and S. Khan, “Efficient 2d graph slam for sparse sensing,” in *IEEE IROS*, 2022.

[6] D. Zou, Y. Wu, and W. Yu, “Structvio: Visual-inertial odometry with structural regularity of man-made environments,” *IEEE T-RO*, 2019.

[7] K. Joo, P. Kim, M. Hebert, I. S. Kweon, and H. J. Kim, “Linear rgb-d slam for structured environments,” *IEEE T-PAMI*, 2021.

[8] E. Jeong, J. Lee, S. Kang, and P. Kim, “Linear four-point lidar slam for manhattan world environments,” *IEEE RA-L*, 2023.

[9] J. M. Coughlan and A. L. Yuille, “Manhattan world: Compass direction from a single image by bayesian inference,” in *IEEE ICCV*, 1999.

[10] G. Schindler and F. Dellaert, “Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments,” in *IEEE CVPR*, 2004.

[11] J. Straub, J. J. Leonard, and J. W. Fisher, III, “A mixture of manhattan frames: Beyond the manhattan world,” in *IEEE CVPR*, 2014.

[12] S. Thrun and M. Montemerlo, “The graph slam algorithm with applications to large-scale mapping of urban structures,” *IJRR*, 2006.

[13] G. Tsai and B. Kuipers, “Real-time indoor scene understanding using bayesian filtering with motion cues,” in *IEEE ICCV*, 2011.

[14] J. Liu and Z. Meng, “Visual slam with drift-free rotation estimation in manhattan world,” *IEEE RA-L*, 2020.

[15] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, “On measuring the accuracy of slam algorithms,” *AURO*, 2009.

[16] K. Murphy and S. Russell, “Rao-blackwellised particle filtering for dynamic bayesian networks,” in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 499–515.

[17] T. N. Yap and C. R. Shelton, “Slam in large indoor environments with low-cost, noisy, and sparse sonars,” in *IEEE ICRA*, 2009.

[18] V. Niculescu, T. Polonelli, M. Magno, and L. Benini, “Nanoslam: Enabling fully onboard slam for tiny robots,” *IEEE IoT*, 2023.

[19] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE T-RO*, 2017.

[20] G. A. Borges and M.-J. Aldon, “A split-and-merge segmentation algorithm for line extraction in 2d range images,” in *IEEE ICPR*, 2000.

[21] M. FISCHLER AND, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, 1981.

[22] M. De Berg, *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

[23] R. Kümmerle, G. Grisetti, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in *IEEE ICRA*, 2011.

[24] M. Yang, Y. Huang, Y.-H. Dai, and B. Li, “Solving heated oil pipeline problems via mixed integer nonlinear programming approach,” *arXiv preprint arXiv:1907.10812*, 2019.

[25] D.-N. Ta, M. Kobilarov, and F. Dellaert, “A factor graph approach to estimation and model predictive control on unmanned aerial vehicles,” in *IEEE ICUAS*, 2014.

[26] A. Howard and N. Roy, “The robotics data set repository (radish),” 2003. [Online]. Available: http://radish.sourceforge.net/

[27] R. T. Rodrigues, N. Tsiogkas, A. Pascoal, and A. P. Aguiar, “Online range-based slam using b-spline surfaces,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1958–1965, 2021.