2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Efficient 2D Graph SLAM for Sparse Sensing

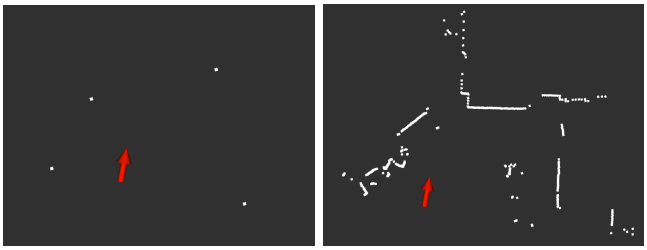
Hanzhi Zhou*,†, Zichao Hu*,†, Sihang Liu†, Samira Khan†

***Abstract*— Simultaneous localization and mapping (SLAM) plays a vital role in mapping unknown spaces and aiding autonomous navigation. Virtually all state-of-the-art solutions today for 2D SLAM are designed for dense and accurate sensors such as laser range-finders (LiDARs). However, these sensors are not suitable for resource-limited nano robots, which become increasingly capable and ubiquitous nowadays, and these robots tend to mount economical and low-power sensors that can only provide sparse and noisy measurements. This introduces a challenging problem called SLAM with sparse sensing. This work addresses the problem by adopting the form of the state-of-the-art graph-based SLAM pipeline with a novel frontend and an improvement for loop closing in the backend, both of which are designed to work with sparse and uncertain range data. Experiments show that the maps constructed by our algorithm have superior quality compared to prior works on sparse sensing. Furthermore, our method is capable of running in real-time on a modern PC with an average processing time of 1/100th the input interval time.**

I. INTRODUCTION

SLAM is the problem of estimating position and orientation while also constructing a map of the environment [1]. Solving SLAM is beneficial to navigation tasks such as path planning, robot recycling, and human decisions. Virtually all state-of-the-art 2D SLAM solutions today are designed for robots with dense and accurate sensors such as laser range-finders (LiDARs). On the contrary, recent work has shown that small, agile, and cheap nano drones demonstrate potential to carry out dangerous indoor exploration missions [2][3]. These nano drones have limited battery and carrying capacity, and it is only possible to mount low-power sensors that can only provide sparse and noisy measurements. For example, as illustrated in Figure 1a, the Crazyflie nano quadrotor [4] can only sense 4 range measurements[1] at 10Hz with a maximum range of 4 meters [5], which is almost two orders of magnitude fewer data compared to a typical 2D LiDAR with more than 180 range measurements and a range of 10 meters, as in Figure 1b. As the result, the limited sensing capacity introduces a challenging SLAM problem.

Prior work [6][7] has shown some progress on overcoming the SLAM with sparse sensing problem. In their work, they adopt the particle filter to solve the problem. However, particle filter has its limitations. First, it cannot refine and globally optimize the complete trajectory of the robot. Furthermore, the number of particles required for a good mapping result becomes increasingly larger when the environment space is large, slowing down computation.



(a) 4 range measurements (b) 180 range measurements

Fig. 1: Sparse (left) vs dense (right) sensing

On the other hand, graph-based techniques have become the standard for modern SLAM solutions because of their superior accuracy, efficiency, and ability to refine the complete trajectory of the robot [8][9]. A specific form of a graph-based technique called pose graph optimization (PGO) has been studied the most in literature because of its simple and sparse structure, allowing it to be solved very efficiently [10].

PGO requires a frontend that computes accurate robot pose-to-pose relations to achieve locally consistent trajectory, typically achieved through scan-matching [11][12]. It assumes that the observations from two consecutive poses have significant overlaps so that a rigid-body transformation can be calculated directly. However, sparse sensing invalidates this assumption. We propose a novel approach called landmark graph that replaces scan-matching as the frontend to address this problem. Instead of calculating a transformation by aligning observations, we utilize the fact that the sparse input accumulated over time can reflect the environment's descriptive structure (landmark). Thus, given the uncertainty of the data, we can form hypotheses of pose-to-pose and pose-to-landmark relations in a graph. Running non-linear least-square optimization updates the hypotheses and calculates a locally consistent trajectory. Our method yields similar accuracy to scan-matching, allowing PGO to be used with sparse range data.

PGO also requires a backend that periodically establishes loop closure constraints between poses that resemble similar places. The current state-of-the-art solution, correlative scan-to-map matching [13], when applied to sparse input data with high uncertainty, tends to fail frequently at differentiating the correct matches from the incorrect ones because their scores are similar. We propose an approximate match heuristic that matches each point in the scan to not a specific cell in the map but a neighborhood of cells. Our experiment shows that the heuristic makes it much easier to set a threshold to differentiate correct and incorrect matches.

*The authors contributed equally.

†Department of Computer Science, University of Virginia, {hz2zz, zh2wc, sihangliu, samirakhan}@virginia.edu

[1]It can also sense the distances to the floor and ceiling, but they cannot contribute to 2D mapping.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981200

To demonstrate the effectiveness of our algorithm, we perform extensive experiments on both our datasets collected with the Crazyflie nano quadrotor [4], and Radish datasets [14]. For Crazyflie datasets, we show that with only 4 range measurements at 10 Hz, we can produce a map very close to the floor plan in the indoor environment. For Radish datasets, we show that our algorithm runs much faster than prior work and produces maps with superior quality. Finally, our parameter sweep experiment indicates that our algorithm can produce reasonable maps with as few as 4 measurements, whereas other mainstream 2D SLAM algorithms fail catastrophically.

To summarize, our contributions are threefold:

- We are the first to provide an open-source[2] graph-based solution to solve the 2D SLAM with sparse sensing problem. Our system is capable of running in real-time on a typical modern computer.
- We propose a novel landmark graph to replace scan-matching as the frontend for sparse range data. The landmark graph can correct relative poses and achieve locally consistent trajectories.
- We propose an approximate match heuristic to the correlative scan-to-map matching algorithm to amplify the score distinction between a correct and an incorrect match, making it easier to reject incorrect loop closures with a threshold.

II. Related Works

Traditional SLAM solutions use filtering approaches such as Extended Kalman Filter [15] and Particle Filter (PF) [16][17]. These approaches maintain poses and landmarks and perform prediction and update steps recursively. Modern works shift more attention toward the optimization approach, often known as graph-based SLAM.

First introduced by Lu and Milios in 1997 [18], the graph-based SLAM approaches model the SLAM problem as a sparse graph of constraints and apply nonlinear optimizations to refine robot trajectory. With the development of efficient and user-friendly backend solver frameworks [19][20][21], graph-based approach excels in accuracy over large space, because of its ability to refine past trajectory [18]. Additionally, advances of robust graph SLAM methods such as Switchable Constraints [22], Dynamic Covariance Scaling (DCS) [23], Max-Mixture [24], and Incremental SLAM with Consistency Checking (ISCC) [25] make graph SLAM resistant to outlier sensor measurements and improve its convergence. Many works show that graph-based SLAM can be used for a variety of sensor configurations. For example, ORB-SLAM2 [26] and RTAB-Map [27] are graph-based systems that specialize in mapping with stereo and RGB-D cameras. Cartographer [28] uses graph-based methods and focuses on real-time mapping using LiDARs.

A subfield of SLAM problems is called SLAM with sparse sensing, in which the robot is limited in sensing capability and can only receive very few data points from the sensors. It is a more challenging problem because the system receives less information with more uncertainty. However, to our knowledge, only a few works have been proposed to solve this problem. Beevers et al. [6] group consecutive observations to extract line features as landmarks and use the Rao-Blackwellized Particle Filter (RBPF) [29] to solve the SLAM problem. Yap et al. [7] utilizes a similar approach to tackle the problem of noisy sonar sensors, but they also applied RBPF while assuming that the walls are orthogonal to each other to produce accurate maps. However, RBPF has limitations – it is not able to refine the past trajectory, and it gets increasingly more computation and memory intensive as the space gets larger because many more particles are required to maintain the accuracy of the map [8].

Our work aims to address the limitations of previous work on SLAM with sparse sensing. We are the first to apply a graph-based approach to this problem. Nevertheless, the adaptation of the graph-based approach is still nontrivial. Due to the lack of sufficient overlapping between different frames of the input data, conventional scan-matching techniques [30] are not applicable, and the already challenging loop closure problem becomes more challenging.

III. Graph SLAM with Sparse Sensing

In order to solve SLAM with sparse sensing, we incorporate two graphs in our approach: landmark graph and pose graph. The landmark graph aims to derive accurate odometry constraints to replace scan matching as the frontend of the pose graph. We leverage the information from a group of scans collected over a period of time, known as 'multiscan', and extract the line segments from the multiscan to describe the environment. Subsequently, we utilize the uncertainty (covariance) of the raw odometry and the line segments to form constraints in the landmark graph.

As for the pose graph, we perform state copying from the landmark to obtain accurate odometry constraints and then apply correlative scan-to-map matching to perform loop closure detection periodically. We propose an approximate match heuristic to sharpen the score distinction between a good and a bad match, thus simplifying the process of finding a threshold. In the following sections, III-A introduces the detailed formulation of the landmark graph, and III-B presents the construction of the pose graph.

Figure 2 illustrates the high-level flow of our approach and shows the need for a special frontend for sparse sensing. It can be observed that when raw odometry is noisy (e.g. not well calibrated), simply building a standard pose graph (a) from raw odometry with correct loop closures is insufficient to achieve a reasonable result. By contrast, we first constructs the landmark graph (b) to obtain a partially corrected estimate of the robot trajectory. Then, we build a pose graph by taking the landmark graph as input along with loop closures, resulting in a map close to the ground truth (c).

A. Landmark Graph

Algorithm 1 shows the high-level procedure for updating and optimizing the landmark graph when a new measure-

[2]https://github.com/shiftlab-nanodrone/sparse-gslam

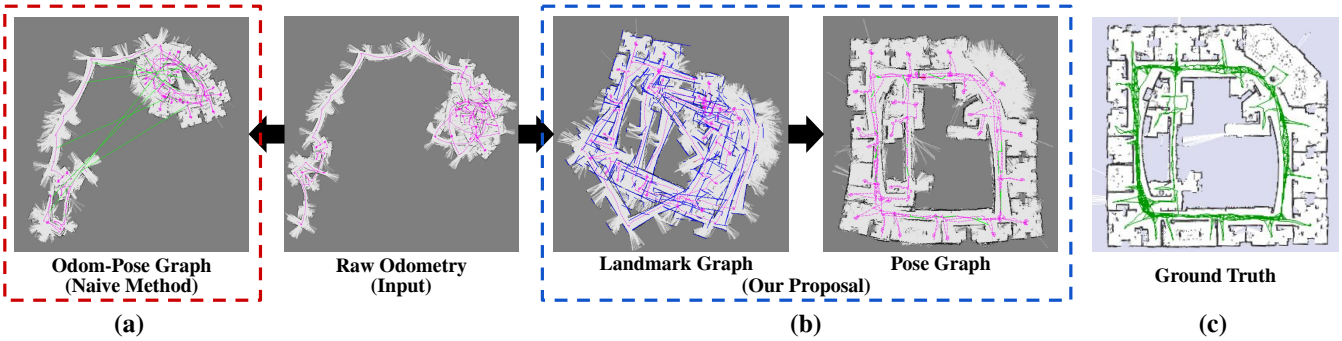


Fig. 2: Illustration of our approach on the Intel Lab dataset. For subfigures except the ground truth, the robot trajectory is in pink, landmark line segments are in blue, and loop closures are in green.

Algorithm 1 Landmark graph update procedure

1: Create a pose vertex and odometry constraint
2: Construct multiscan
3: Extract line segments from multiscan
4: **for** each segment in segments **do**
5: Associate line segments
6: Create a constraint and insert into graph
7: **end for**
8: Save graph's state
9: Optimize landmark graph
10: **if** Graph is inconsistent **then**
11: Remove the newly inserted constraints
12: Restore graph's state
13: **else**
14: Update each line segment's endpoints
15: **end if**

ment arrives from the sensor. The details of each line in Algorithm 1 will map to the sections below.

1) Notation: Let the state of the robot at time t to be $\mathbf{x}_t = (x, y, \theta)^T$ and the control input to be $\mathbf{u}_t = (\Delta x, \Delta y, \Delta\theta)^T$. Then, the next robot state $\mathbf{x}_{t+1}$ can be obtained using the standard motion composition operator $\oplus$ (see section 3.2 of [31]). The observation in the coordinate frame of $\mathbf{x}_t$ is denoted as $\mathbf{z}_t^t$, where the superscript is the index of the frame of reference and the subscript is the observation index. For 2D range measurements, $\mathbf{z}_t^t \in \mathcal{R}^{2\times n}$ represents n 2D Cartesian coordinates, which we will refer as a 'scan'.

2) Multiscan construction (line 2): Similar to prior work [6], we form a multiscan from the observations at multiple robot poses. For real-time SLAM systems, it is unreasonable to incorporate future observations, because it will cause delay in processing. Thus, we choose to construct a multiscan from k previous scans: $\mathbf{z}_t, \mathbf{z}_{t-1}, ..., \mathbf{z}_{t-k}$. Define a transformation function g on a 2d point $\mathbf{p} = (a, b)$ by a pose $\mathbf{x} = (x, y, \theta)$.

$$g(\mathbf{p}, \mathbf{x}) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \mathbf{p} + \begin{pmatrix} x \\ y \end{pmatrix} \quad (1)$$

In order to assemble a multiscan in the frame of $\mathbf{x}_t$, we need to transform each of the previous observations, $\mathbf{z}_{t-i}, i \in [0, k]$, to the frame of x_t:

$$\mathbf{z}_{t-i}^t = g(g(\mathbf{z}_{t-i}^{t-i}, \mathbf{x}_{t-i}), \mathbf{x}_t^{-1}) \quad (2)$$

Suppose that $\mathbf{x}_t$ is the consequence of the robot motion $\mathbf{u}_{t-i}, ..., \mathbf{u}_{t-1}$. Define

$$\mathbf{u}_{it} = \mathbf{u}_{t-i} \oplus \mathbf{u}_{t-i+1} ... \oplus \mathbf{u}_{t-1} \quad (3)$$

Therefore, $\mathbf{x}_t = \mathbf{x}_{t-i} \oplus \mathbf{u}_{it}$. Then, it can be shown that

$$\mathbf{z}_{t-i}^t = g(\mathbf{z}_{t-i}^{t-i}, \mathbf{u}_{it}^{-1}) \quad (4)$$

We will use (4) to transform the observations from the previous frames to the current frame.

3) Line segment extraction (line 3): We implement split-and-merge to extract line segments from each multiscan, because it has the best trade-off between efficiency and accuracy as shown by Nguyen et al. [32]. Each line is represented in polar form: $\mathbf{l} = (\rho, \alpha)$ where $\rho >= 0$ and $\alpha \in [-\pi, \pi)$ for its compactness, and we refer the reader to Garulli et al. [33] for the details this representation and least-square line fitting.

4) Line segment association (line 4-7): To determine if a currently observed segment is a part of an existing landmark with parameter (ρ, α), we use two criteria:

1) The line projection error is smaller than a threshold
2) Projected endpoints sufficiently overlap with the global line segment

Given the endpoints $\mathbf{p}_1, \mathbf{p}_2$ of the observed segment in the global frame, criterion 1 can be formulated as

$$\|\mathbf{p}_1 - (\mathbf{a} + t_1\mathbf{d})\| + \|\mathbf{p}_2 - (\mathbf{a} + t_2\mathbf{d})\| \leq \varepsilon \quad (5)$$

where

$$\mathbf{a} = (\rho\cos\alpha, \rho\sin\alpha)^T, \ \mathbf{d} = (-\sin\alpha, \cos\alpha)^T \quad (6)$$

$$t_1 = (\mathbf{p}_1 - \mathbf{a}) \cdot \mathbf{d}, \ t_2 = (\mathbf{p}_2 - \mathbf{a}) \cdot \mathbf{d} \quad (7)$$

To test criterion 2, we first project the endpoints of the landmark onto itself with (7) to get two scalar values s_1 and s_2. Then, criterion 2 is satisfied when

$$[s_1, s_2] \cap [t_1, t_2] \neq \emptyset \quad (8)$$

For all landmarks that satisfy (5) and (8), the one with the smallest projection error is associated with the current observation. If no landmarks satisfy both criteria, a new landmark is created for future associations.

5) Graph optimization (line 9): We formulate the objective function as

$$F(X) = \sum_i \|e_o(\mathbf{x}_i, \mathbf{x}_{i+1})\|^2_{\Sigma_i} + \sum_{ij} \|e_l(\mathbf{x}_i, \mathbf{l}_j)\|^2_{\Sigma_{ij}} \quad (9)$$

where e_o is the error function of the odometry constraints

$$e_o(\mathbf{x}_i, \mathbf{x}_{i+1}) = \mathbf{v}^{-1}_{i,i+1} \oplus (\mathbf{x}_i^{-1} \oplus \mathbf{x}_{i+1}) \quad (10)$$

where $\mathbf{v}_{i,i+1}$ is the odometry measurement between the two poses. e_l is the error of the pose-landmark constraints[3]

$$e_l(\mathbf{x}_i, \mathbf{l}_j) = \mathbf{v}_{ij} - f(\mathbf{x}_i^{-1}, \mathbf{l}_j) \quad (11)$$

where $\mathbf{v}_{ij} = (\rho_{ij}, \alpha_{ij})$ is the measurement of the landmark $\mathbf{l}_j$ seen in the frame of $\mathbf{x}_i$, and $f(\mathbf{x}_i^{-1}, \mathbf{l}_j)$ transforms the current estimate of landmark $\mathbf{l}_j$ from the global frame to the frame of $\mathbf{x}_i$, which is given by[4]

$$f(\mathbf{x}, \mathbf{l}) = \begin{pmatrix} \rho + x\cos(\alpha+\theta) + y\sin(\alpha+\theta) \\ \text{normAngle}(\alpha+\theta) \end{pmatrix} \quad (12)$$

To calculate covariances of odometry constraints Σ_i in (9), we employ first-order error propagation. Given control inputs that cause the robot to move from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$

$$\mathbf{x}_{i+1} = \mathbf{x}_i \oplus \mathbf{u_1} \oplus ... \oplus \mathbf{u_n} \quad (13)$$

the covariance can be calculated recursively as

$$\begin{aligned} \Sigma_i &= \text{Cov}(\mathbf{u}_1 \oplus ... \oplus \mathbf{u_n}) \\ &= J_\oplus \left[\begin{array}{c|c} \text{Cov}(\mathbf{u}_1) & \mathbf{0} \\ \hline \mathbf{0} & \text{Cov}(\mathbf{u}_2 \oplus ... \oplus \mathbf{u_n}) \end{array}\right] J_\oplus^T \end{aligned} \quad (14)$$

where $J_\oplus$ is the Jacobian of the motion composition operator $\oplus$ with respect to its inputs. To calculate the covariances of pose-landmark constraints Σ_{ij} in (9), we assume points $\mathbf{p}_k$ that constitute the given landmark observation are uncorrelated, and therefore

$$\Sigma_{ij} = \sum_k J_k \,\text{Cov}(\mathbf{p}_k)\, J_k^T \quad (15)$$

where J_k is the Jacobian of the least-square line fitting function with respect to each point $\mathbf{p}_k$ whose expression is provided by Garulli et al. [33]. Since each point is transformed by (4) during multiscan construction, their covariances can be approximated as

$$\text{Cov}(\mathbf{p}_k) = J_g \left[\begin{array}{c|c} \text{Cov}(\mathbf{u}_{it}^{-1}) & \mathbf{0} \\ \hline \mathbf{0} & \text{Cov}(\mathbf{p}) \end{array}\right] J_g^T \quad (16)$$

where J_g is the Jacobian of g w.r.t. its inputs, and

$$\text{Cov}(\mathbf{u}_{it}^{-1}) = J_{\mathbf{u}_{it}^{-1}} \text{Cov}(\mathbf{u}_{it}) J^T_{\mathbf{u}_{it}^{-1}} \quad (17)$$

where $\text{Cov}(\mathbf{u}_{it})$ can be calculated in a way similar to (14). $\mathbf{Cov}(\mathbf{p})$ is original source of error of the observation, which can be modeled as

$$\text{Cov}(\mathbf{p}) = \sigma_d^2 \begin{bmatrix} \cos(\theta)^2 & \sin(\theta)\cos\theta \\ \sin(\theta)\cos\theta & \sin(\theta)^2 \end{bmatrix} \quad (18)$$

[3]The error of angle α needs to be normalized to $[-\pi, \pi)$ range.

[4]We need to make sure that the landmark after this transform has $\rho >= 0$. If not, the angle needs to be incremented by π and normalized again.

Algorithm 2 Line segment endpoint update procedure

```
1: for each landmark do
2:     Initialize s_1 = ∞, s_2 = −∞
3:     Calculate vector representation a, d with (6)
4:     for each observation of this landmark do
5:         Calculate t_1, t_2 of the observed segment with (7)
6:         s_1 = min(s_1, t_1), s_2 = max(s_2, t_2)
7:     end for
8:     Calculate the new endpoints: a + s_1 d, a + s_2 d
9: end for
```

Algorithm 3 Pose graph update procedure

```
1: Detect loop closures
2: if best match score > threshold then
3:     Copy the state of the landmark graph to pose graph
4:     Prune landmark graph
5:     Insert loop closure constraint with DCS kernel
6:     Optimize pose graph
7: end if
```

where θ is the bearing of $\mathbf{p}$ that is assumed to have no error and σ_d is the error of this range measurement.

After all constraints for the current observations are inserted into graph, graph optimization is performed. The optimal solution $X^* = \text{argmin}(F(X))$ of (9) is solved by g2o [19] with the Levenberg-Marquardt solver.

6) Consistency checking (line 10-12): Since it is possible for line association to produce incorrect matches that introduce significant errors to the graph, we follow the idea of ISCC [25] and implement a simplified version. Assuming the noise of the errors follows Gaussian distribution, $F(X)$ follows χ^2 distribution, so we can check whether

$$F(X) \leq \chi^2(0.95, n) \quad (19)$$

where $\chi^2(\cdot,\cdot)$ is the inverse chi-squared CDF and n is the sum of degrees of freedom of all constraints. If (19) is not satisfied, then at least one of the constraints inserted in the current batch is an outlier. To ensure performance, they are all discarded and are not checked one by one.

7) Line segment endpoint update (line 14): The endpoints information is essential to compute (8). When a pose-landmark constraint is created, the endpoints of the observed segment are stored in addition to the line parameters. The algorithm to update line endpoints is shown in Algorithm 2.

B. Pose Graph

The purpose of the pose graph is to produce a globally consistent map with the initial estimate from the landmark graph and the help of loop closures. The high level procedure for updating the pose graph is shown in Algorithm 3.

1) Notation: To distinguish the vertices of the pose graph from the landmark graph, $\mathbf{y}_i$ is used to represent these vertices. Each $\mathbf{x}_i$ in the landmark graph provides initial estimate for $\mathbf{y}_i$ in the pose graph. $\mathbf{w}_{i,i+1}$ is the odometry measurement between $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$, which is derived from

the landmark graph as

$$\mathbf{w}_{i,i+1} = \mathbf{x}_i^{-1} \oplus \mathbf{x}_{i+1} \tag{20}$$

2) Loop closure detection and approximate match heuristic (line 1-2): We mainly follow the approach of the Cartographer [28]. First, after every certain distance traveled, we create a local submap using the combined observations during this interval. Then, the submaps are stored as occupancy grids, in which each cell stores a probability of it being occupied. At the same time, we continuously construct multiscan from several recent poses[5] to match against all previous submaps using correlative scan-to-map matching [13], defined as finding the transformation $\mathbf{w}^*$ that best aligns the scan h with submap M.

$$\mathbf{w}^* = \arg\max_{\mathbf{w}\in\mathcal{W}} \sum_{i=1}^{N} M\lfloor \mathbf{w}h_i \rceil \tag{21}$$

where $\mathcal{W}$ is the search space for the transformation, $\mathbf{w}h_i$ transforms the ith range measurement in the scan to the submap coordinate frame, and $M\lfloor\cdot\rceil$ returns the occupancy of the nearest grid cell. If the sum in (21) for $\mathbf{w}^*$ is greater than a threshold, we will accept the loop closure as a constraint in the pose graph.

Nevertheless, both the multiscan and the map are noisy due to the sparsity of the input data. This frequently causes the algorithm to consider matches with points off by a few centimeters as bad matches, making it harder to differentiate good from bad matches effectively. A naive way to solve the problem is to use a larger cell size for the submap grid. However, this negatively affects the scan-to-map matching accuracy due to the decrease in submap resolution.

To solve the problem, we use an approximate match heuristic. Instead of using the nearest cell for each point in the scan to calculate a score, we apply a 3x3 max kernel around the grid cell corresponding to each observed point.

$$\mathbf{w}^* = \arg\max_{\mathbf{w}\in\mathcal{W}} \sum_{i=1}^{N} \max_{x,y\in[-1,1]} M\lfloor \mathbf{w}h_i + (x,y)^T \rceil \tag{22}$$

The effect of applying this max kernel is illustrated in Figure 3. Before applying the kernel, both the correct and incorrect matches have similar scores, making it hard to set a threshold hold to distinguish them. After applying the kernel, the score difference between them is significant enough to distinguish them effectively.

Although (22) seems computational expensive due to the repeated application of the 3x3 max kernel, in reality, one can precompute the submap occupancy grid with the kernel applied because submaps are fixed after construction. Therefore, our method (22) have no performance penalty compared to the original method (21) besides a small submap initialization cost.

[5]These poses need to have their estimates adjusted in the landmark graph before combining the sparse range measurements associated with each of them

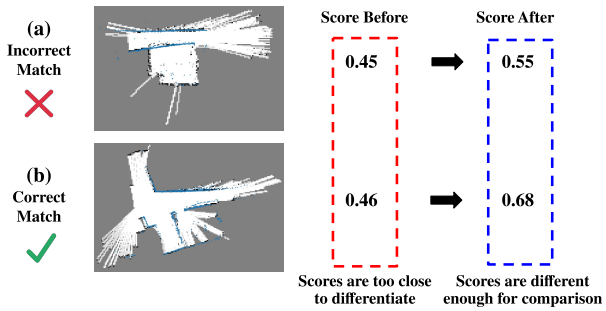


Fig. 3: Blue points are observations to be matched with the submaps. Applying the max kernel makes it easier to distinguish the good match (b) from the bad match (a).

3) Graph state copying and pruning (line 3-4): To make use of the optimized pose estimates in the landmark graph, the odometry constraints between consecutive pose vertices are calculated with (20). Additionally, the estimates of the poses in the pose graph are copied from the landmark graph after each successful loop closure. Note that only the estimates of the poses inserted after the last loop closure optimization are copied. This ensures that we keep the estimates of vertices that are already optimized.

After the state is copied, the landmark graph can be pruned by fixing the recently added vertices and edges and removing all other edges and constraints. This pruning procedure ensures that the landmark graph's size remains small, thus making it efficient even in prolonged exploration tasks.

4) Graph optimization (line 4-5): The pose graph objective follows the classical formulation as introduced by Sünderhauf et al. [34], which is given by

$$F(X) = \sum_i \|e_o(\mathbf{y}_i, \mathbf{y}_{i+1})\|^2_{\Sigma_i} + \sum_{ij} \|e_{lc}(\mathbf{y}_i, \mathbf{y}_j)\|^2_{\Sigma_{ij}} \tag{23}$$

where e_o is identical to (10), and

$$e_{lc}(\mathbf{y}_i, \mathbf{y}_j) = \mathbf{w}_{ij}^{-1} \oplus (\mathbf{y}_i^{-1} \oplus \mathbf{y}_j) \tag{24}$$

where $\mathbf{w}_{ij}$ represents a rigid transformation between pose $\mathbf{y}_i$ and $\mathbf{y}_j$ calculated by the loop closure detector. In (23), the covariance matrices for odometry constraints, Σ_i, are copied from the landmark graph. The covariance matrices for loop closure constraints, Σ_{ij}, are calculated by fitting a Gaussian distribution to the neighborhood of $\mathbf{w}_{ij}$ as in Olson [13].

However, the least-square formulation of graph SLAM by itself is not resistant to outlier constraints, which may arise due to uncertainty and ambiguity of measurement and the environment. Sparse sensing makes this problem worse by providing less measurement with more uncertainty. Hence, robust SLAM methods are critical to ensure the success of our algorithm. A few robust SLAM methods can work with our model, such as ISCC [25], and DCS [23]. As per our experiments, DCS works the best in our implementation, and it is fairly easy to tune. Finally, we used Gauss-Newton algorithm provided by g2o [19] to minimize (23).

IV. Experiments

To demonstrate the effectiveness of our algorithm, we perform extensive experiments on the well-established Radish

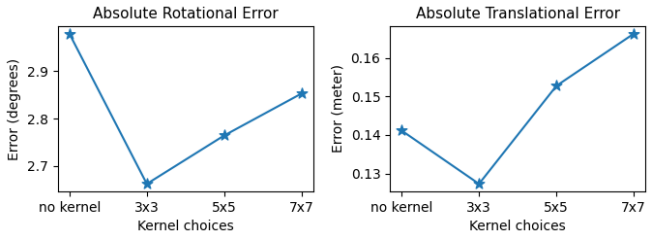


Fig. 4: Kernel choices vs mapping error

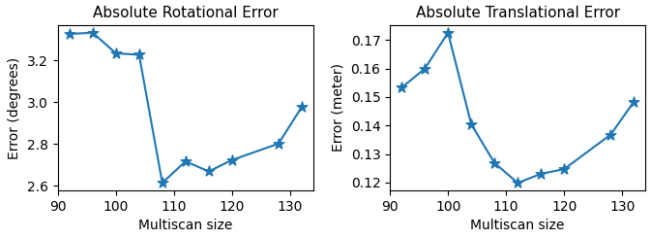


Fig. 5: Multiscan sizes vs mapping error

[14] datasets and real-world sparse sensing datasets we collected. The quality of maps are measured by *absolute translational error* and *absolute rotational error*, proposed by Kümmerle et al. [35] who also provide the ground truth for the datasets. When not otherwise specified, the grid cell size for our maps used in the experiment is 0.1m. 4 range measurements are sampled from each scan to simulate sparsity, and each range measurement is capped at 5m. Multiscan size is 120, meaning that 30 consecutive 4-point scans are grouped as the input for the landmark graph.

A. Comparison among approximate matching kernels

To justify the use of the 3x3 kernel in (22), we compare the quality of the maps constructed from the Intel Lab dataset using no kernel, 3x3 kernel, 5x5 kernel, and 7x7 kernel. As shown in Figure 4, 3x3 kernel is the ideal size that provides the appropriate fuzziness against sparse and noisy data, yielding the lowest mapping error. Using larger kernel sizes will harm matching accuracy due to the significantly increased tolerance for mismatches.

B. Comparison among multiscan sizes

For our proposed landmark graph to work, an appropriate multiscan size needs to be chosen. As shown in Figure 5, although our algorithm can work with a wide range of multiscan sizes, there is a ”sweet spot” in the middle which gives low errors. The reason is that when the multiscan size is too small, fewer line segments can be extracted, causing the frontend’s accuracy to degrade. On the other hand, when the multiscan size is too large, errors from odometry will accumulate more, causing the extracted line segments to tilt away from the ground truth. The middle range provides good balances between data density and error accumulation.

C. Sensitivity study on the number of range measurements

Although our algorithm is designed for sparse sensing, it will be interesting to see how the algorithm performs as measurements get denser. In Figure 6, we vary the number

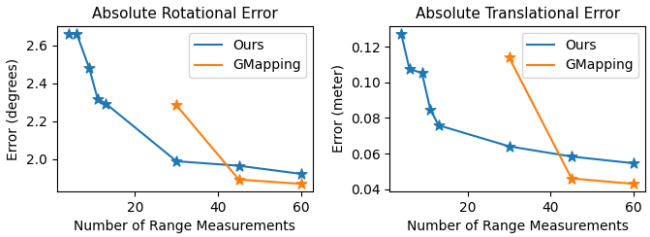


Fig. 6: Number of range measurements vs mapping error

of range measurements sampled from the Intel Lab dataset from 4 to 60 (out of 180). For reference, we also present the results of GMapping [36].[6]

We can observe that the mapping accuracy improves with the addition of new range measurements. While GMapping fails under 30 range measurements, our method can still produce reasonable maps, even when there are as few as 4 range measurements per scan.

Note that when measurements get denser, diminishing return can be observed. The explanation for diminishing return is that our frontend is tailored for sparse data by interpolating probable line segments, and it does not benefit directly and significantly from the additional characteristics exhibited by the increasing density of data.

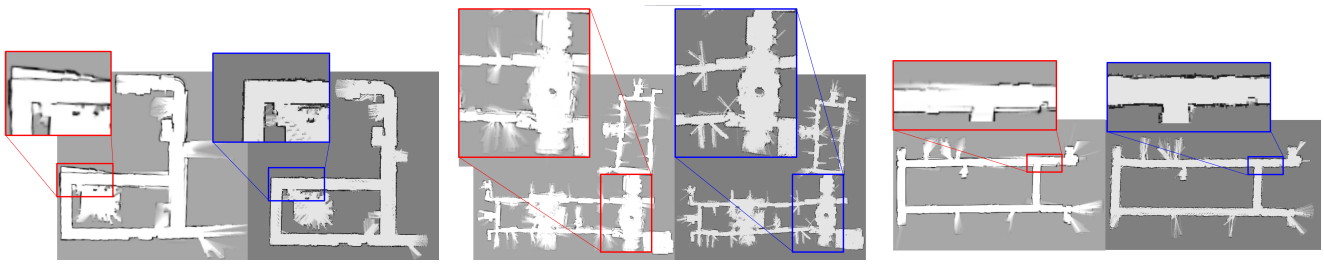
D. Comparison with previous works

We first run our algorithm on the datasets used by Beevers et al. [6] We follow their way to simulate sparse sensing – sub-sample range measurements and set distance cap – for a fair comparison. As shown in Figure 7, our algorithm has more solid and clear representation of the walls. Furthermore, unlike previous work, our algorithm is less susceptible to spurious landmarks. For example, we highlight a few places in Figure 7 where the previous work has noticeable aliasing while we do not.

Second, we evaluate our approach on Aces, Intel Lab, and MIT Killian datasets from Radish. Even for a SLAM system with dense input data, it is challenging to produce good results on these datasets because (1) Intel Lab’s odometry is extremely noisy, (2) Aces lacks revisits of the same places, which makes loop closure more challenging, and (3) MIT Killian dataset has a considerable dimension of roughly 190m by 240m. Since no quantitative baselines for sparse sensing exist, we have no choice but to compare our results with GMapping [36], a dense sensing method.

In Table I we present results of GMapping with 30 range measurements (30pt), which is the lowest number that can yield a good map for each dataset. While our method can work with as few as 4 range measurements, we present results using 11 range measurements (11pt), which can produce results as least comparable to the results of GMapping across all datasets. As we can observe, our proposal can produce results comparable to or better than GMapping with much

[6]We also experimented Hector SLAM [37] and Cartographer [28]. However, as they are designed for dense sensing (e.g., via modern LiDARs), they do not produce meaningful output under moderately sparse settings (e.g., less than 60 range measurements).



(a) CMU NSH (b) Stanford Gates (c) USC SAL

Fig. 7: For each subfigure, the left image is from prior work [6] while the right image is our result. We highlight the visual details to demonstrate the better quality of our generated maps.

TABLE I: Quantitative comparison with GMapping

	This Work	GMapping
Aces	11pt	30pt
Absolute translational	0.0455 ± 0.0492	0.1040 ± 0.2839
Absolute rotational	1.159 ± 1.440	1.334 ± 2.421
Intel Lab	11pt	30pt
Absolute translational	0.0848 ± 0.1151	0.1139 ± 0.2274
Absolute rotational	2.319 ± 2.371	2.283 ± 2.331
MIT Killian	11pt	
Absolute translational	0.0718 ± 0.1913	Fail to produce map
Absolute rotational	2.065 ± 3.846	

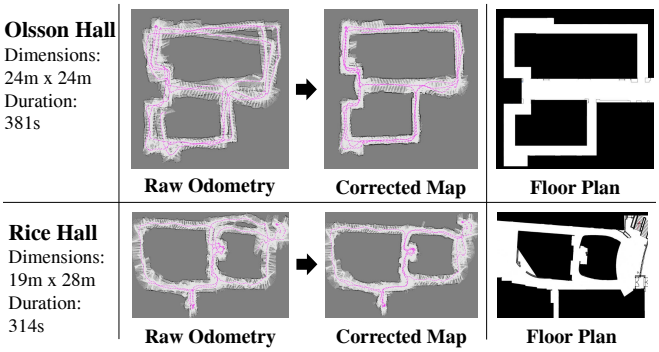


Fig. 8: Maps built with data from Crazyflie compared with the floor plan. Drone trajectory is in pink. The irrelevant details of the floor plan are covered in black.

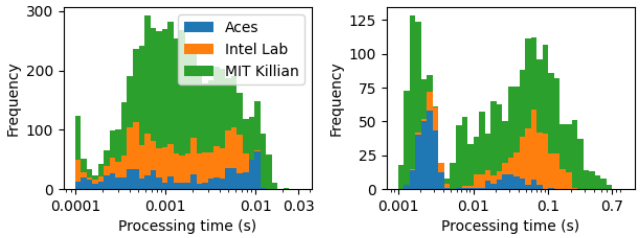
fewer range measurements. Additionally, GMapping fails to complete the MIT Killian dataset within a reasonable amount of time due to the large dataset size and high number of particles needed to produce good maps under sparse settings.

E. Real world sparse sensing datasets

Besides using sub-sampling to simulate sparse sensing, we demonstrate our work on real-world robots with sparse range sensors. We utilized the Crazyflie [4] nano-quadrotor, which weighs only 27g and is capable of estimating its trajectory with its IMU and PMW3901 optical flow sensor. We equip it with 4 VL53L1x ToF sensors providing distance to the front, back, left, and right, with an effective range of 2m and at a rate of 10Hz. We collect around 5 to 6 minutes of flight data in Olsson Hall and Rice Hall of University of Virginia by driving the drone around manually. As shown in Figure 8, although the drone can localize itself with dead-reckoning to some extent, the odometry error still accumulates over time, leading to map aliasing. Our algorithm can correct the map and produce results similar to the floor plan. Since the range measurements are incredibly sparse, it is worth noting that some regions of the occupancy grid are not completely filled.

TABLE II: Speed evaluation. Unit is in seconds

Dataset	ACES	Intel Lab	MIT Killian
Average data interval	**0.185**	**0.197**	**0.439**
Mean processing time	0.0008	0.0022	0.0061
Max frontend processing time	0.011	0.011	0.021
Max backend processing time	0.106	0.214	0.654
GMapping mean processing time	0.348	0.240	Fail



(a) Frontend (b) Backend

Fig. 9: Processing time distribution. Time is in log scale.

F. Speed Evaluation

To demonstrate the real-time capability of our algorithm, we evaluate our system's processing time on a Desktop PC with Intel Core i7-9700K. The frontend (Algorithm 1) runs synchronously with the input data, while the backend (Algorithm 3) runs asynchronously. We summarize the key statistics in Table II and the time distribution as a stacked histogram in Figure 9. It can be observed that our method use only 1/100th of the data interval on average with most processing happens very quickly. This can be attributed to the efficient frontend and graph pruning. Occasional processing time peaks in frontend are also well below the average data interval. For the backend, its processing time allows it to execute a few times per second in the background, allowing loops to be closed very promptly.

On the other hand, GMapping spends significant amount of time for each dataset, due to need for many particles for map to be reasonably accurate. This shows why graph-based approaches are more suited for sparse sensing than particle based methods used in prior works.

V. Conclusions

This paper presented the first graph-based system to address challenging SLAM with sparse sensing problems. The solution incorporated a novel frontend analogue to scan matching but tailored for sparse sensing and an improved loop closure detection algorithm. Our system is evaluated using various datasets, and it shows promising ability to handle even large real-world indoor exploration tasks. Possible future directions of research include extending the algorithm to solve the multi-robot SLAM with sparse sensing.

ACKNOWLEDGMENT

Radish datasets [14] are used to benchmark our algorithm. Thank Patrick Beeson, Dirk Hähnel, Mike Bosse, John Leonard, Andrew Howard, Nick Roy, and Brian Gerkey for providing these datasets.

References

[1] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[2] S. Li, C. D. Wagter, and G. C. H. E. de Croon, "Self-supervised monocular multi-robot relative localization with efficient deep neural networks," *CoRR*, vol. abs/2105.12797, 2021. [Online]. Available: https://arxiv.org/abs/2105.12797

[3] B. P. Duisterhof, S. Krishnan, J. J. Cruz, C. R. Banbury, W. Fu, A. Faust, G. C. H. E. de Croon, and V. J. Reddi, "Learning to seek: Autonomous source seeking with deep reinforcement learning onboard a nano drone microcontroller," *CoRR*, vol. abs/1909.11236, 2019. [Online]. Available: http://arxiv.org/abs/1909.11236

[4] 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 *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2017, pp. 37–42.

[5] *A new generation, long distance ranging Time-of-Flight sensor based on ST's FlightSense technology*, STMicroelectronics, 11 2018, rev. 3.

[6] K. Beevers and W. Huang, "SLAM with sparse sensing," in *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2285–2290.

[7] T. N. Yap and C. R. Shelton, "SLAM in large indoor environments with low-cost, noisy, and sparse sonars," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1395–1401.

[8] D. Wilbers, C. Merfels, and C. Stachniss, "A comparison of particle filter and graph-based optimization for localization with landmarks in automated vehicles," in *Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 220–225.

[9] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[10] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 22–29.

[11] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[12] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 19–25.

[13] E. B. Olson, "Real-time correlative scan matching," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 4387–4393.

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

[15] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986. [Online]. Available: https://doi.org/10.1177/027836498600500404

[16] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association," vol. 2004, 2004.

[17] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," *Proc. IJCAI Int. Joint Conf. Artif. Intell.*, 06 2003.

[18] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *AUTONOMOUS ROBOTS*, vol. 4, pp. 333–349, 1997.

[19] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[20] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[21] V. Ila, L. Polok, M. Solony, and P. Svoboda, "Slam++ 1 -a highly efficient and temporally scalable incremental slam framework," *The International Journal of Robotics Research*, vol. 36, pp. 210 – 230, 2017.

[22] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1879–1884.

[23] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 62–69.

[24] E. Olson and P. Agarwal, *Inference on Networks of Mixtures for Robust Robot Mapping*. MIT Press, 2013, pp. 313–320.

[25] M. C. Graham, J. P. How, and D. E. Gustafson, "Robust incremental SLAM with consistency-checking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 117–124.

[26] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[27] M. Labbé and F. Michaud, "RTAB-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831

[28] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[29] A. Doucet, N. d. Freitas, K. P. Murphy, and S. J. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, ser. UAI '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, p. 176–183.

[30] A. Mallios, "Sonar scan matching for simultaneous localization and mapping in confined underwater environments," in *University of Girona Thesis*, 2014.

[31] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," 2013.

[32] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1929–1934.

[33] A. Garulli, A. Giannitrapani, A. Rossi, and A. Vicino, "Mobile robot SLAM for line-based environment representation," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 2041–2046.

[34] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph SLAM," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 1254–1261.

[35] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, pp. 387–407, 11 2009.

[36] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[37] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155–160.