Efficient Graph SLAM for Sparse Sensing

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

***Abstract*— Small, agile, and cheap nano drones demonstrate promising potentials to carry out dangerous indoor exploration missions, in which simultaneous localization and mapping (SLAM) plays a vital role in mapping out unknown spaces and aiding autonomous navigation. However, state-of-the-art solutions for SLAM are tailored for dense and accurate sensors such as laser range-finders (LiDARs), which are uneconomical to be mounted on nano drones due to their limited battery and carrying capacity. Although some priors work has addressed SLAM with sparse sensing problems, they use filtering-based approaches that cannot solve the full SLAM problem. They are either making strong assumptions about the environment or computationally expensive. This work presents a system that uses line segment features with raw odometry to derive constraints in a pose graph, which are subsequently optimized with loop closures. Our experiments show that our method can construct 2D maps with 100 times real-time performance. Furthermore, the maps constructed by our algorithm have superior quality compared to prior works on sparse sensing and are comparable to maps produced with dense sensing.**

I. INTRODUCTION

Recent work has shown that small, agile, and cheap nano drones demonstrate potentials to carry out dangerous indoor exploration missions [1][2]. SLAM is the problem of estimating position and orientation while also constructing a map of the environment [3]. Solving SLAM is beneficial to navigation tasks such as path planning and drones recycling, and it can also aid human decisions. However, nano drones' limited battery and carrying capacity make it only economical to mount low-power sensors that can only provide sparse and noisy measurements. For example, the Crazyflie nano quadrotor [4] can only sense 4[1] range measurements 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 a 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, the particle filtering-based approach has its limitations. It cannot refine the complete trajectory of the robot, and the number of particles needed to keep track of becomes increasingly larger when the environment space is large. On the other hand, with recent advances in SLAM, graph-based optimization techniques have become the standard for the most modern 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 has been studied the most in literature because of its simple and sparse structure, allowing it to be solved very efficiently [10].

There are two types of constraints in the pose graph: odometry constraints and loop closure constraints. Odometry constraints are formed between two consecutive poses, while loop closure constraints are formed between any two poses resembling a similar place. To produce odometry and loop closure constraints, variants of scan-matching techniques [11][12][13] are used. They rely on the fact that when many data are available, the observations from two poses have significant overlap, and consequently, a rigid-body transformation can be calculated by aligning these observations. However, sparse sensing data hardly exhibit overlapping characteristics, making these techniques not applicable.

Our work aims to address the limitations of scan-matching techniques on sparse data and adapt the pose graph formulation to solve the SLAM with sparse sensing problem. In order to construct a pose graph from the sparse data, we first need to derive reasonable odometry constraints. We take inspiration from previous work [6] to combine observations from several consecutive poses to form a multiscan to increase the density of data. Subsequently, we extract line segments as landmarks from the multiscan, and we associate them to form spatial constraints between poses and the landmark. We make improvements by deriving a systematic covariance propagation from observation to line segment parameters. Finally, we put these constraints and the raw odometry together to formulate a new graph, which we refer to as the landmark graph. When we optimize the landmark graph, the corrected pose relations can be obtained to produce odometry constraints for the pose graph, thus achieving similar effects as the scan-matching techniques.

Second, to establish loop closure constraints in the pose graph, we employ multiscan again on the scan-to-map matching algorithm [14], which periodically builds maps from observations as the robot moves and matches the robot's current observation with previous maps. However, since the multiscan agglomerates observations from different poses, the noise adds up, which renders the multiscan more uncertain and deteriorates the algorithm's ability to detect a good loop closure. We alleviate this problem by proposing an approximate match heuristic, where each point in the multiscan is not matched to a specific cell in the map but a neighborhood of cells. This simple heuristic aims to sharpen the distinction between a good and a bad match, and the

*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.

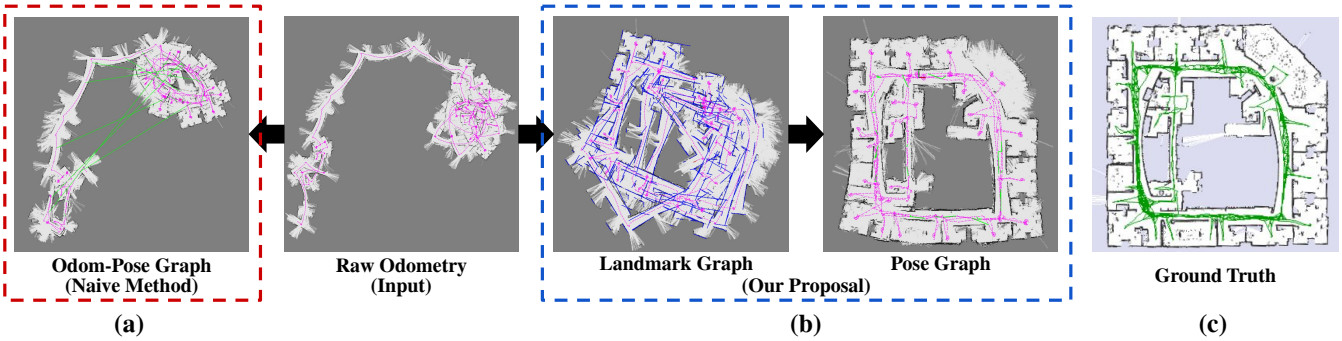


Fig. 1: Illustration of our approach on the Intel-Lab dataset. For subfigures except the ground truth, the robot trajectory is in pink, landmark line segments are in blue, and loop closures are in green.

experiment shows the heuristic works great in practice to differentiate the quality of the matches.

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 [15]. 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 compare our results with previous work on sparse sensing [6] and show that the maps constructed by our algorithm are visibly superior. We also show that our maps are comparable to the maps produced with dense sensing approaches, such as Cartographer [11].

To summarize, our contributions are threefold:

- We propose a technique for sparse sensing that utilizes multiscans and line features to derive odometry constraints for the pose graph.
- We address the challenging loop closure detection problem by presenting a novel approximate match heuristic.
- We perform extensive experiments on both the real-world sparse sensing datasets and the well-established Radish datasets.

II. Related Works

Traditional SLAM solutions use filtering approaches such as Extended Kalman Filter [16] and Particle Filter (PF) [17][18]. 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 [19], 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 [20][21][22], graph-based approach excels in accuracy over large space, because of its ability to refine past trajectory [19]. Additionally, advances of robust graph SLAM methods such as Switchable Constraints [23], Dynamic Covariance Scaling (DCS) [24], Max-Mixture [25], and Incremental SLAM with Consistency Checking (ISCC) [26] 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 [27] and RTAB-Map [28] are graph-based systems that specialize in mapping with stereo and RGB-D cameras. Cartographer [11] 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. A few notable works have been proposed to solve this problem. Beevers et al. group consecutive observations to extract line features as landmarks and use the Rao-Blackwellized Particle Filter (RBPF) [29] to solve the SLAM problem [6]. Yap et al. tackle the problem of noisy sonar sensors [7]. They also apply RBPF with an assumption 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 the SLAM with sparse sensing problem, we incorporate two graphs in our approach: landmark graph and pose graph. We build the landmark graph from the extracted line features and the raw odometry measurements. The goal of the landmark graph is to derive more accurate odometry constraints than raw odometry so that we can subsequently use these constraints to construct the pose graph. The reason to build a pose graph and not other graph formulation is that robust methods for the pose graph are studied more extensively in the literature and are readily available. As for the loop closure problem in the pose graph,

we propose a novel approximate match heuristic to address the challenging loop closure problem.

Figure 1 illustrates the high-level flow of our approach. It can be observed that the raw odometry is very noisy. Therefore, even with correct loop closures, simply building a standard pose graph (a) from raw odometry 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).

In the following sections, III-A introduces the detailed formulation of the landmark graph and III-B presents the construction of the pose graph.

A. Landmark Graph

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
```

In the landmark graph, the estimates of the robot's poses and the line features are represented as vertices. The constraints (edges) are odometry constraints, which form between two consecutive poses, and pose-landmark constraints, which form between poses and the landmarks (line segments) observed by these poses.

Algorithm 1 shows the high-level procedure for updating and optimizing the landmark graph when a new measurement 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) Whether the reprojection error as in Ruifang et al. [34] is smaller than a threshold
2) Whether the 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 reprojection 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

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

where e_o is the error function of the odometry constraints

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

where $\mathbf{v}_{i,i+1}$ is the odometry measurement between the two poses. e_l is the error of the pose-landmark constraints[2]

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

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[3]

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

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 $\mathbf{x}_{i+1}$

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

the covariance can be calculated recursively as

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

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

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

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

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

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

$$\text{Cov}(\mathbf{u}_{it}^{-1}) = J_{\mathbf{u}_{it}^{-1}} \text{Cov}(\mathbf{u}_{it}) J_{\mathbf{u}_{it}^{-1}}^T \tag{17}$$

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

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

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 [20] 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, a reduced version of ISCC [26] is implemented. Assuming the noise of the errors follows Gaussian distribution, $F(X)$ follows χ^2 distribution, so we can check

$$F(X) \leq \chi^2(0.95, n) \tag{19}$$

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.

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

[3]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:     Insert loop closure constraint with DCS kernel
5:     Optimize pose graph
6: end if
```

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 [11]. 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 to match against all previous submaps using correlative scan-to-map matching [14]. Finally, a score is evaluated for each match, and we define a threshold to accept or reject the loop closure.

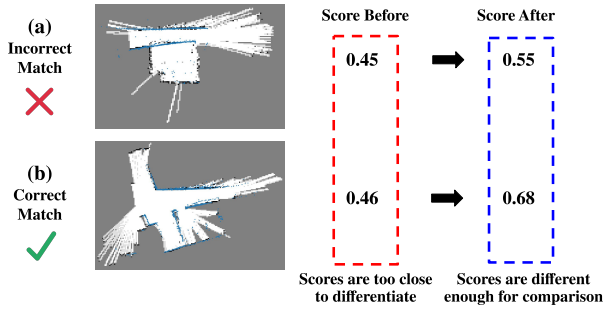


Fig. 2: Blue points are observations to be matched with the submaps. Applying the max kernel makes it easier to distinguish the good match (b) from the bad match (a).

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 effectively differentiate good from bad matches as illustrated in Figure 2. To solve the problem, we use an approximate match heuristic. Instead of averaging the scores of all observed points to the map grid, we use a 3x3 max kernel around the grid cell corresponding to each observed point to calculate its score. Experiment results show that this simple heuristic is effective in separating incorrect matches from correct ones.

3) Graph State Copying (line 3): 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. 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.

4) Graph Optimization (line 4-5): The pose graph objective follows the classical formulation as introduced by Sünderhauf et al. [35], which is given by

$$F(X) = \sum_i \|e_o(\mathbf{y}_i, \mathbf{y}_{i+1})\|^2_{\Sigma_i} + \sum_{ij} \|e_{lc}(\mathbf{y}_i, \mathbf{y}_j)\|^2_{\Sigma_{ij}} \quad (21)$$

where e_o is identical to (10), and

$$e_{lc}(\mathbf{y}_i, \mathbf{y}_j) = \mathbf{w}_{ij}^{-1} \oplus (\mathbf{y}_i^{-1} \oplus \mathbf{y}_j) \quad (22)$$

where $\mathbf{w}_{ij}$ represents a rigid transformation between pose $\mathbf{y}_i$ and $\mathbf{y}_j$ calculated by the loop closure detector. In (21), 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 a neighborhood of the rigid transformation estimated by the loop closure detector as formulated by Olson [14].

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 [26], and DCS [24]. 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 [20] to minimize (21).

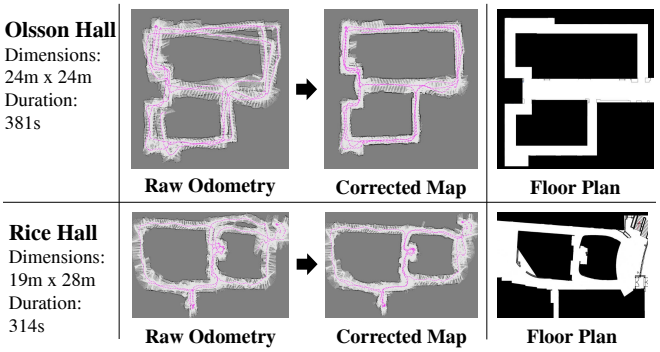


Fig. 3: Maps built with data from Crazyflie compared with the floor plan. The irrelevant details of the floor plan are covered in black. Also note that floor plans do not include dynamic objects such as tables and chairs.

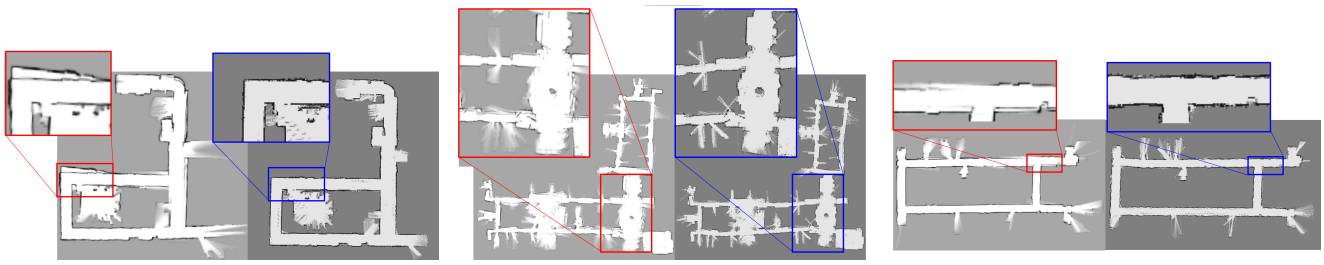
IV. Experiments

We extensively evaluate our algorithm on a variety of datasets. In subsection IV-A, we present real-world results using data collected by us with Crazyflie [4] in Olsson Hall and Rice Hall at the University of Virginia (UVa). In subsection IV-B, we run our algorithm on the established Radish [15] datasets. Unfortunately, no quantitative baseline has been presented for the SLAM sparse sensing, so we take the following approaches to demonstrate the effectiveness of our algorithm. First, we compare our map qualitatively against prior work [6]. Second, we calculate quantitative metrics formulated by Kümmerle et al. [36] and compare our results with Cartographer, because it is a well-established work for using 2D range measurements. Note that this comparison is at our disadvantage as we use a magnitude fewer data. Finally, in subsection IV-C, we compare the execution time of our algorithm with sensor data duration to demonstrate the efficiency of our algorithm.

A. Crazyflie Datasets

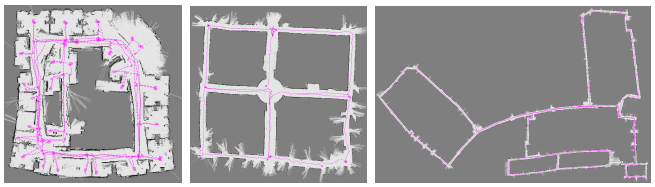
The Crazyflie nano quadrotor weighs only 27g, and it is capable of estimating its trajectory with its IMU and PMW3901 optical flow sensor. We equip it with 4 VL53L1x ToF sensors providing distance to the front, back, left, and right obstacles, at a rate of 10Hz[4]. We collect data in Olsson Hall and Rice Hall by driving the drone around manually. Due to limited battery capacity, Crazyflie can only sustain about 5 minutes of flight, so it can only explore a limited area. As shown in Figure 3, although the drone can localize itself with dead-reckoning to some extent, the odometry error still accumulates over time, leading to map aliasing. Nevertheless, our algorithm can correct the map and produce results similar to the floor plan. Since the range measurements are extremely sparse, it is worth noting that some regions of the occupancy grid are not completely filled.

[4]Although the sensor can provide a maximum rate of 50Hz, to achieve reliable $\geq$ 2 meter sensing range, data rate must be lowered to 10Hz.



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

Fig. 4: 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.



(a) Intel-Lab (b) Aces (c) MIT Killian Court

Fig. 5: Results of our algorithm on Radish datasets. Robot trajectory is in pink. (a) and (b) show great visual results, and (c) demonstrates potentials to produce a reasonable map in the presence of large loops.

B. Radish Datasets

We first run our algorithm on the datasets used by prior work [6], where we take the same sub-sampling rate and distance cap for range measurements for a fair comparison. As shown in Figure 4, 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 highlights a few places in Figure 4 where the previous work has noticeable aliasing while we don't.

Second, we evaluate our approach on Aces, Intel-Lab, and MIT-Killian datasets from Radish. We sample only 11 range measurements out of each 180-degree laser scan to simulate sparse sensing. As shown in Figure 5, we can produce great visual results for Intel-Lab and Aces and reasonable results for MIT-Killian. However, even for a SLAM system with dense input data, it is challenging to produce good visual 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. Hence, our results demonstrate the potentials to handle complex exploration tasks in large indoor areas. As far as we know, the largest public dataset that prior works on SLAM with sparse sensing can cover is the Stanford-Gates, which has a dimension of roughly 64m by 56m. On the other hand, our approach can run on MIT-Killian, almost one magnitude larger than the state-of-the-art.

We also quantitatively compare our metrics against the Cartographer as shown in Table I. Even though our algorithm receives one magnitude less data than the Cartographer and the comparison is at our disadvantage, the experiment shows that our results are reasonable with only a few centimeters in translational error and one to two degrees of rotational error.

TABLE I: Quantitative Comparison with Cartographer

	Ours	Cartographer [11]
Aces		
Absolute translational	0.0463 ± 0.0510	0.0375 ± 0.0426
Absolute rotational	1.179 ± 1.463	0.373 ± 0.469
Intel-Lab		
Absolute translational	0.0884 ± 0.1295	0.0229 ± 0.0239
Absolute rotational	2.456 ± 2.570	0.453 ± 1.335
MIT-Killian		
Absolute translational	0.0676 ± 0.1474	0.0395 ± 0.0488
Absolute rotational	2.048 ± 3.750	0.352 ± 0.353

TABLE II: Speed Evaluation

Dataset	Data duration (s)	Wall clock (s)
USC SAL	488.2	1.7
CMU NSH	593.1	1.3
Stanford Gates	2077.0	34.8
ACES	1365.5	7.2
Intel Lab	2691.3	35.5
MIT Killian	7676.6	81.7

C. Speed Evaluation

To demonstrate the efficiency of our algorithm, we compare the wall clock time against the duration of sensor data. Our algorithm is run on Intel Core i7-9700K, a modern desktop PC CPU. As shown in Table II, our algorithm can process the sensor data very efficiently, using only 1/100th of the data duration on average. Hence, our algorithm demonstrates the potential to process inputs in real-time.

V. Conclusions

This paper presented a novel graph-based system to address the scan matching and loop closure challenges in the graph SLAM with sparse sensing problems. The system is evaluated using various datasets, and it shows potentials to handle 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 [15] 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] 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

[2] 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

[3] 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.

[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] 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.

[12] 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.

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

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

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

[16] 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

[17] 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," *Journal of Machine Learning Research*, vol. 2004, 2004.

[18] 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.

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

[20] 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.

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

[22] V. Ila, L. Polok, M. Solony, and P. Svoboda, "Highly efficient compact pose SLAM with SLAM++," *CoRR*, vol. abs/1608.03037, 2016. [Online]. Available: http://arxiv.org/abs/1608.03037

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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

[29] A. Doucet, N. de Freitas, K. P. Murphy, and S. J. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," *CoRR*, vol. abs/1301.3853, 2013. [Online]. Available: http://arxiv.org/abs/1301.3853

[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] D. Ruifang, V. Fremont, S. Lacroix, I. Fantoni, and L. Changan, "Line-based monocular graph SLAM," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2017, pp. 494–500.

[35] 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.

[36] 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.