CG-SLAM: Cooperative Graph-based SLAM*

Geoffrey H. Harrison[1]

***Abstract*— Simultaneous Localization and Mapping (SLAM) is a common problem in robotics with a variety of solutions, such as Extended Kalman Filter, pose graph optimization, and particle filter methods. In SLAM, robots construct a map of their environment and place themselves within the map, aided by measurements from the robot's sensors. Noise in sensory data can introduce error into the resulting map. This project attempts to reduce map error by combining information from multiple robots performing SLAM simultaneously. We implement an approach that combines a pose graph optimization approach to the SLAM problem with merging multiple maps across robots to achieve a shared map with lower robot pose and landmark error than the robots could achieve individually, called CG-SLAM. CG-SLAM is empirically validated through use of a dataset of measurements and ground truth from multiple robots operating simultaneously.**

I. INTRODUCTION

Simultaneous Localization and Mapping, or SLAM, is the problem of a robot obtaining a spatial map of its environment and simultaneously localizing the robot's position within the map. A critical and difficult problem for building autonomous mobile robots, it has seen many solutions developed, though these solutions often struggle with mapping dynamic or large environments. [1] Solving SLAM requires the use of the robot's sensors to determine both the map of the environment and the robot's pose. However, sensors commonly used in mobile robotics systems, such as cameras (especially of the monocular variety), odometry sensors, LiDAR, and radar, may give noisy or otherwise unreliable measurements, introducing error to any resulting map.

The reliability of the robot's localization, key to navigating successfully and safely, depends greatly on the accuracy of the estimated map features, especially in contexts such as navigation of autonomous vehicles where failure could have severe conseqeuences. It is therefore important, especially in the presence of noisy sensory data, to reduce map error as much as possible. In cases where multiple robots operate simultaneously within the same environment, such as using "swarms" of UAVs collaboratively mapping a large area, or a fleet of autonomous vehicles traversing the same road, allowing robots to exchange their information may produce a more accurate map, reducing the amount of error present in each robot's map.

This project considers the case of performing SLAM across multiple robots in a decentralized fashion. By allowing robots to exchange and merge their accumulated maps as they meet, each robot's map comes to contain more observations of the landmarks than they could achieve on their own, providing more constraints on map optimization to obtain a better estimate of landmark location. Simulating this method using a pre-collected dataset, we evaluate the resulting maps for accuracy of both landmark locations and robot poses.

*This work was not supported by any organization

[1]Geoffrey H. Harrison is with the Institute for Aerospace Studies, University of Toronto, Toronto, ON M5S, Canada ghharrison@cs.toronto.edu

II. RELATED WORK

Approaches to SLAM are broadly classified in three groups: Extended Kalman Filter methods, which approach SLAM as an online problem; graph-based methods, which solve "full SLAM" using offline algorithms; and particle filter methods, sampling particles according to an approximation of a posterior distribution of state. [1] EKF methods have somewhat fallen out of favor in recent literature, as their computational properties make them difficult to scale efficiently. Particle filter methods such as FastSLAM, on the other hand, are known to suffer from particle deprivation; as particle filter methods work by calculating the posterior probability of the measurement against each particle in the map, if new information on a previously observed landmark differs considerably from the existing map, it will have a very low weight as an "unlikely" hypothesis and thus may not be sampled at all. [2] In the context of multi-robot SLAM, this makes it difficult to update landmark poses in light of new observations shared from other robots.

Pose graphs represent both the robot's motion and the environment in the form of a graph, where the vertices represent robot poses or environment landmarks, and the edges linking them represent sensory measurements, such as from robot odometry or camera data. Each measurement edge imposes a soft constraint upon the final arrangement of the graph's vertices, and violation of these constraints is minimized through an optimization step. [3] As no sampling is used for pose graph methods, in contrast to particle filter methods, it is well-suited to represent a map that may be re-optimized repeatedly as previously discovered landmarks are re-observed from additional robots.

Classical approaches to the problem involve only a single robot, yet various methods have been developed to allow SLAM to be performed cooperatively across multiple robots. Morrison, Gavez-Lopez, and Sibley present a multi-robot SLAM implementation in which robots communicated their data to a server that performs map fusion, offloading the work of map-merging from the robots to improve scalability. [4] Similarly, Gil, Reinoso, Ballesta, and Julia use a visual particle-filter method where maps are merged from the combination of visual descriptors computed from landmark observations and robot odometry measurements. [5] Deutsch,

Liu, and Siegwart demonstrate a method for performing pose-graph SLAM across multiple robots using visual features for loop closure. [6] However, these methods all share in common a *centralized* approach, requiring a connection from all robots to a central point of contact. Should the connection from each robot to the center become unreliable, or should the central controller not exist, robots will struggle to benefit from other robots' mapping.

Decentralized techniques for multi-robot SLAM also exist. Menon describes a particle filter-based variation on FastSLAM that works across a swarm of robots, additionally replacing broken robots in an automatic and decentralized fashion. [7] However, as with the underlying FastSLAM, particle deprivation may result in accumulation of pose error over time. Leung, Barfoot, and Liu introduce an algorithm that performs SLAM under a network of robots using ORB-SLAM visual features to detect landmarks, where the network is constantly changing and not fully connected. [9] They further define conditions in which a "centralized-equivalent estimate" of robot state and landmark position can be obtained by all robots using an EKF-based approach. However, these approaches are not based on pose graphs, and are unable to take advantage of graph-based SLAM's desirable properties.

We here seek to perform pose-graph SLAM in a decentralized manner, but we approach this via a somewhat simpler problem in that landmarks can be reliably and uniquely identified from sensor data without needing to incorporate visual feature extraction into the algorithm. Our objective is to develop a method where the pose graphs may be optimized and exchanged between multiple robots in order to minimize landmark and robot pose error.

III. METHODOLOGY

A. Overview

To implement our form of graph-based SLAM, we closely follow the notation of Grisetti et al. [3] In the common formulation of graphical SLAM, due to sensory noise poses and measurements are taken as random variables. Formally, we have a robot trajectory represented as a sequence of random variables $x_{1:T} = \{x_1, ..., x_T\}$, where T is the maximum number of timesteps, a sequence of odometry measurements $u_{1:T} = \{u_1, ..., u_T\}$, and landmark measurements $z_{1:T} = \{z_1, ..., z_T\}$. The objective of SLAM is to estimate the posterior probability of the robot's trajectory and a map of the environment, conditioned on the measurements as well as an initial position:

$$p(x_{1:T}, m | z_{1:T}, u_{1:T}, x_0) \tag{1}$$

(If the ground truth initial position x_0 is not known, we can arbitrarily initialize it at the origin $(0, 0)$.) We here represent the map the set of landmark positions, but other representations such as point clouds or occupancy grids are also possible depending on the choice of sensors equipping the robot. In our case, we have the somewhat simpler objective of, rather than computing an estimate of the posterior probability, to find an estimated trajectory $\hat{x}_{1:T}$ and map of landmarks $\hat{m}$ that are most likely given the odometry and sensory measurements; the maximum likelihood estimate. [3]

B. Assumptions

To perform our simulation, we use the Multi-Robot Collaborative Localization and Mapping (MR.CLAM) datasets, a group of two-dimensional datasets of robot odometry measurements, range-and-bearing measurements from a monocular camera, and ground-truth data for robot and landmark positions over time. [10] Subsequently, motion is restricted to a two-dimensional space, and range and bearing estimates to robots and landmarks visible through each robot's camera are readily available. We use a discrete time simulation with timesteps of fixed length dt; during each timestep, we assume that forwards and angular velocity do not change from their values at the start of the timestep. That is, $v_{t+\Delta t} = v_t$ for time t if $\Delta t < dt$ and $\omega_{t+\Delta t} = v\omega_t$ if $\Delta t < dt$.

We additionally define a maximum communication range over which robots can exchange their maps; if the difference in ground-truth poses between two robots attempting to merge maps exceeds this range, the communication fails and no changes are made to either robot's map. Subsequently, the network formed by robots capable of exchanging maps is not fully connected at all times.

Optimizing the map requires an information matrix associated with each edge, representing the certainty of the edge's measurement. This is often the inverse of the covariance matrix of sensor measurements at each observation. [3] MR.CLAM datasets do not include estimates of sensor measurements, but comparing the dataset's odometry measurements to the actual ground-truth change in pose at each timestep (of length $dt = 1\text{s}$), we observed that the odometry readings were generally accurate to a level of $(0.05\text{m}, 0.05\text{m}, 0.001\text{rad})$, allowing us to approximate the information matrix for odometry edges[1] as

$$\begin{aligned}\Omega &= \begin{bmatrix} 0.05^2 & 0 & 0 \\ 0 & 0.05^2 & 0 \\ 0 & 0 & 0.001^2 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 400 & 0 & 0 \\ 0 & 400 & 0 \\ 0 & 0 & 100 \end{bmatrix}\end{aligned} \tag{2}$$

Similarly, camera measurements were often accurate to roughly 0.5 meters estimating the distance to the landmark, but due to accumulating pose error the bearing measurement varied quite a lot, so the information matrix for landmark observations[1] was taken as

$$\begin{aligned}\Omega &= \begin{bmatrix} 0.5^2 & 0 & 0 \\ 0 & 0.5^2 & 0 \\ 0 & 0 & \left(\frac{\pi}{2}\right)^2 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0.406 \end{bmatrix}\end{aligned} \tag{3}$$

[1]These information matrices are different from those used when obtaining the results for the video presentation accompanying this report. Subsequently, the reported results are different from those of the video as well.

As these are diagonal matrices, we implicitly assume that measurement errors in each dimension are independent. This may not necessarily be the case in reality; an inaccurate bearing measurement will cause the calculated x- and y-coordinates of the landmark to be off as well even if the measured range is correct.

C. State Estimation

Poses of all robots and landmarks are represented in $\mathcal{SE}(2)$ form, consisting of an x-coordinate and y-coordinate (measured in meters), and an orientation θ (measured in radians). Landmarks are trivially considered to always have an orientation of zero. Camera observations consist of a subject identifier, a range k in meters to target, and the bearing of the target $\theta_{bearing}$ in the robot's local frame. The x- and y-coordinates of an observed landmark can be determined by calculating the relative position $\hat{x}_{landmark}, \hat{y}_{landmark})$ of the landmark in the robot's local frame and composing it with the robot's global pose (x, y, θ):

$$\hat{x}_{landmark} = k\cos(\theta_{bearing}) \tag{4}$$

$$\hat{y}_{landmark} = k\sin(\theta_{bearing}) \tag{5}$$

$$x_{landmark} = x + \hat{x}_{landmark}\cos(\theta) - \hat{y}_{landmark}\sin(\theta) \tag{6}$$

$$y_{landmark} = y + \hat{x_{landmark}}\sin(\theta) + \hat{y}_{landmark}\cos(\theta) \tag{7}$$

$$\theta_{landmark} = 0 \tag{8}$$

Odometry measurements consist of a forwards velocity v (measured in the robot's local frame) and an angular velocity ω. Assuming that the same v and ω hold for the entire duration of the timestep dt, the new position and orientation of the robot at time $t + dt$ can be calculated as:

$$\begin{aligned} x_{t+dt} = {} & r\cos(\omega \cdot dt)sin(\theta) \\ & + r\sin(\omega \cdot dt)cos(\theta) \\ & + x_t - r\sin(\theta) \end{aligned} \tag{9}$$

$$\begin{aligned} y_{t+dt} = {} & r\sin(\omega \cdot dt)\sin(\theta) \\ & - rcos(\theta)\cos(\omega \cdot dt) \\ & + y_t + r\cos(\theta) \end{aligned} \tag{10}$$

$$\theta_{t+dt} = \theta_t + \omega dt \tag{11}$$

where r is the radius of the instantaneous circle of curvature associated with the movement. As the change in angle over dt and the distance travelled are known, obeying the arc length formula $s = r \cdot \theta$, r can be calculated as $r = v/(\omega \cdot dt)$.

Prior to map optimization, vertices are recorded once and are not updated again; additional observations of the same landmark will create new edges incident on the landmark node, but will not update its pose. Therefore, in the absence of optimization, the map assumes the first observation of each landmark to be the "correct" one. Observations robots make of other robots are not stored in this implementation; as all poses are represented in $\mathcal{SE}(2)$, and robots cannot estimate other robots' orientations purely from observing them, we opt to discard these measurements rather than attempt to record an arbitrary orientation for other robots.

D. Pose Graph Optimization

Again we use notation introduced by Grisetti et al. [3] In the pose graph, edges are defined by a pair of vertices (x_i, x_j), the measurement between them z_{ij} (the estimated displacement/angle change for odometry vertices, the estimated vector to a landmark for landmark observations), and an information matrix Ω_{ij} that corresponds to the certainty of the measurement. Landmark vertices are unique in the graph; multiple observations of the same landmark introduce additional edges but no new vertices. Subsequently, the estimated measurement may be inconsistent with the "expected measurement" $\hat{z_{ij}}$, the actual difference between vertices as currently represented in the graph. The error function corresponding to the edge linking x_i and x_j is then:

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z_{ij}}(x_i, x_j) \tag{12}$$

(Note that this does not consider the *true* pose of the second vertex in the environment; the ground-truth pose is unavailable to the robot and the "real" pose error cannot be calculated.) Accordingly, we can take the log likelihood of the observation z_{ij} as proportional to

$$l_{ij} \propto [z_{ij}\hat{z_{ij}}(x_i, x_j)]^T \Omega_{ij} [z_{ij}\hat{z_{ij}}(x_i, x_j)] \tag{13}$$

This gives rise to a nonlinear least-squares problem; the "optimized" map will be the maximum likelihood estimate of the true map, or alternatively, the map with its vertices adjusted so as to minimize the negative log likelihood of all recorded observations. This is expressed as

$$F(x) = \sum_{i,j} e_{ij}^T \Omega_{ij} e_{ij} \tag{14}$$

$$x^* = \arg\min_x F(x) = \arg\min_x \sum_{i,j} e_{ij}^T \Omega_{ij} e_{ij} \tag{15}$$

We directly implement nonlinear least squares optimization to achieve the optimal x^*. Given an initial guess $\breve{x}$ of robot and landmark poses, Gauss-Newton optimization is used to obtain at an approximate optimizer. Using first-order Taylor expansion, we write

$$\begin{aligned} e_{ij}(\breve{x_i} + \Delta x_i, \breve{x_j} + \Delta x_j) &= e_{ij}(\breve{x} + \Delta x) \\ &\approx e_{ij}(\breve{x}) + J_{ij}\Delta x \end{aligned} \tag{16}$$

where J_{ij} is the Jacobian matrix of $e_{ij}(\breve{x})$.

$$F_{ij}(\breve{x} + \Delta x)$$

$$= e_{ij}(\breve{x} + \Delta x)^T \Omega_{ij} (e_{ij}(\breve{x} + \Delta x) \tag{17}$$

$$\approx (e_{ij}(\breve{x}) + J_{ij}\Delta x)^T \Omega_{ij} (e_{ij}(\breve{x}) + J_{ij}\Delta x) \tag{18}$$

$$\begin{aligned} = {} & (e_{ij}(\breve{x}))^T \Omega_{ij} (e_{ij}(\breve{x})) + 2(e_{ij}(\breve{x}))^T \Omega_{ij} J_{ij} \Delta x \\ & + (\Delta x)^T J_{ij}^T \Omega_{ij} J_{ij} \Delta x \end{aligned} \tag{19}$$

We define the notation $c_{ij} = (e_{ij}(\breve{x}))^T \Omega_{ij} (e_{ij}(\breve{x}))$, $b_{ij} = (e_{ij}(\breve{x}))^T \Omega_{ij} J_{ij}$, and $\mathbf{H}_{ij} = J_{ij}^T \Omega_{ij} J_{ij}$. This lets us express

Equation 14 as

$$F(\breve{x} + \Delta x) = \sum_{i,j} F_{ij}(\breve{x} + \Delta x) \tag{20}$$

$$\approx \sum_{i,j} c_{ij} + 2b_{ij}\Delta x + (\Delta x)^T \mathbf{H}_{ij}(\Delta x) \tag{21}$$

Setting $c = \sum c_{ij}$, $b = \sum b_{ij}$, and $\mathbf{H} = \sum H_{ij}$, we obtain the quadratic form

$$F(\breve{x} + \Delta x) = c + 2b^T(\Delta x) + (\Delta x)^T \mathbf{H}(\Delta x) \tag{22}$$

This quadratic form has a minimizer $(\Delta x)^*$ that is the solution to

$$\mathbf{H}(\Delta x)^* = -b \tag{23}$$

An optimal solution can be found using Cholesky decomposition on $\mathbf{H}$. There is a possible pitfall, however, in the fact that we represent poses in $\mathcal{SE}(2)$, meaning that the pose space is non-Euclidian, violating an assumption of Gauss-Newton optimization and potentially leading to sub-optimal solutions. [3] In any event, as the optimizer is approximate, the optimization process may be repeated until convergence to find an estimated optimizer for $(\Delta x)^*$. After each iteration, the graph's vertices are updated according to $x_i^* = x_i + (\Delta x)^*$ for all i before optimizing again.

E. Map Merging

The other major component of our approach to SLAM requires that robots cooperate in their mapping and localization by sharing maps with each other in a decentralized fashion. Suppose two robots numbered i and j "meet" by arriving within a certain distance of each other. These robots possess maps $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$. We can express the shared map $G_{i \leftrightarrow j}$ as:

$$G_{i \leftrightarrow j} = (V_i \cup V_j, E_i \cup E_j) \tag{24}$$

When a robot receives a map from another robot, it adds to the receiver's map all vertices and edges from the sender's pose graph that the receiver does not already possess in its map. Robot pose vertices across all robots are uniquely identified by the combination of their creator robot's ID number and the timestep at which they were created, while landmark vertices are identified by their unique ID number as recorded in the measurement that first observed it. Edges are identified by the tuple of vertices they link; as a robot cannot measure the same landmark multiple times simultaneously, measurements from a robot to a landmark at a particular time are unique.

In practice, this means that for any landmarks the first robot has already observed (or received from another robot), the other robot's estimated landmark poses are simply discarded. However, any edges from the other robot will be added to the graph, as they impose additional constraints on the landmark's pose and will aid in optimizing the map, so the exchange is still beneficial for arriving at a better pose estimate for the landmark.

The map merge operation becomes more complicated in the event that robots were not initialized with the same global frame of reference (e.g., robots are not initialized with their ground-truth pose and assume their initial pose to be at the coordinate system origin with an orientation of zero.) The two robots sharing maps must decide on one global frame that both will use and translate into that shared frame. As individual range and bearing measurements may be noisy, robots reach a consensus on their relative positions before attempting to align their maps.

Suppose two robots A and B observe each other, measuring each other as being separated by vectors $m_{\overrightarrow{AB}}$ and $m_{\overrightarrow{BA}}$ respectively (as expressed in each robot's global frame.) In the absence of sensor noise, these vectors should be of equal magnitude. Otherwise, a consensus may be reached by averaging them; in expectation this will yield the true distance between the two robots. [8] If A and B share a global frame, then we should have $m_{\overrightarrow{AB}} = -m_{\overrightarrow{BA}}$ after rescaling. Therefore, the angular difference between the robots' global frames should correspond to a rotation matrix $\mathbf{R} \in \mathbb{R}_{2\times 2}$, where

$$\mathbf{R} \cdot m_{\overrightarrow{BA}} = -m_{\overrightarrow{AB}} \tag{25}$$

Solving for $\mathbf{R}$ yields the angle by which B's map must be rotated to match A's global frame orientation. After applying the rotation about B's current pose (to preserve distance from its pose to A), B can then translate its map so that A's self-reported pose in its frame matches B's estimate of its pose, finally aligning the two maps in a shared frame and allowing map-merging to occur as usual. Since this form of pose consensus requires the two robots to estimate each other's poses, we somewhat modify the earlier stated conditions for when robots may exchange maps: in addition to being within the maximum communication range of each other, *robots only exchange maps when they observe each other simultaneously.* Furthermore, to encourage all robots to eventually align on the same global frame, we can introduce a rule to determine which robot should transform its map to match the other's global frame. A convenient method is to use ID numbers: when robots exchange maps, the robot with the larger ID must transform its map into the other's frame.

With this, we arrive at the final formulation for this approach to multi-robot SLAM, dubbed Cooperative Graph-based SLAM, or CG-SLAM.

IV. EXPERIMENTAL RESULTS

A. Simulation Dataset

CG-SLAM is evaluated by performing a simulation using the UTIAS MR.CLAM datasets. Each of these datasets consists of timestamped odometry measurements for five robots and timestamped range and bearing estimates measured by each of these robots for fifteen landmarks, as well as ground-truth pose information for each. Each robot and landmark is uniquely identified by a barcode that the robots read from their camera observations as they record measurements. As the datasets do not innately contain measurements for all robots at all times, a script was used (modified from that provided with the dataset) to interpolate between dataset entries to ensure that ground-truth and odometry data are

present for all timesteps. (The length of each timestep is a user-provided parameter.) A screenshot from an animation of the ground truth of Dataset 2 is included in Figure 1.

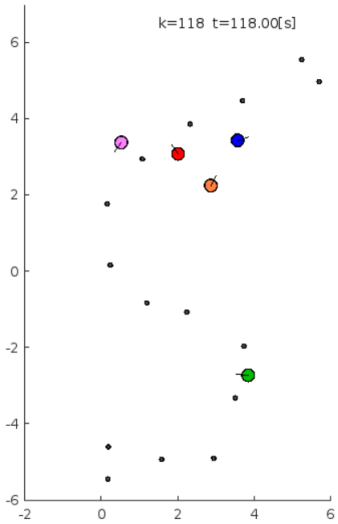

Fig. 1. A screenshot of the ground truth from MR.CLAM Dataset 2. The landmarks are depicted in black while robot poses and orientations can be observed from the colored circles.

Each robot ground-truth file includes a sequence of 2D positions and pose orientations, as well as the time at which each was recorded. Landmarks do not have an orientation, but are understood by the simulation to always have a ground-truth and estimated orientation of 0 radians in the global frame. Robot observation files consist of range and bearing measurements to other subjects (robots and landmarks) as well as the subject number identified with each one, while robot odometry files record the robot's estimated forwards velocity (in m/s) and angular velocity (in radians) at various times.

B. Simulating SLAM

The simulation parses the dataset selected for performing SLAM, and accepts a parameter for the number of active robots. (If fewer than five robots are used, the simulation simply ignores data for the inactive robots.) It then parses the rows corresponding to each timestep, simulating the robots performing a random walk in the mapping environment, updating their positions as estimated by each robot's odometry for that step. After each step, if the simulation finds that two robots can observe each other and are within the maximum communication range (according to ground-truth positions), it has each robot share its map with the other, merging as described in Methodology. Every so often, according to another parameter, the simulation re-optimizes all robots' maps. Finally, after all timesteps are completed, the simulation computes the average landmark and pose error for each robot's accumulated map. Taking $\hat{x}^{(i)}_{1:T}$ as the estimated set of robot i's poses from its graph, the true poses as $x^{(i)}_{1:T}$, and m as the estimated landmark poses, this loss function is calculated as

$$L(\hat{x}^{(i)}_{1:T}) = \frac{\sum_{k=1}^{T} \|x_k^{(i)} - x_k^{(i)}\|_2}{T} \tag{26}$$

$$L(m) = \frac{\sum_{m_k \in m} \|m_k^* - m_k\|_2}{T} \tag{27}$$

Note that each robot does not consider the pose error of any robot other than itself. Each robot measures its estimated pose error separately. Additionally, the landmark loss function assumes that all landmarks are eventually discovered by all robots, an assumption that holds for the MR.CLAM dataset used for our evaluation.

C. Results and Analysis

To ensure that experimental results could be directly compared, all evaluation was performed using MR.CLAM Dataset 2. For each number of robots between 1 and 5, the CG-SLAM simulation was performed using a timestep length of 1 second, a maximum communication range of 2 meters, with optimization every 250 seconds (if optimization is applied at all). Simulations were repeated with these parameters with initial ground-truth robot poses known, so as to ensure that all robots used a common global frame. (If this is not done, the choice of translation for the accumulated map to compare against the ground-truth map may affect the resulting error values.) Results were recorded in three execution modes: without optimization or sharing, with only optimization, and with optimization and sharing. When maps are not optimized, the constraints imposed by the edges are not utilized, effectively meaning that all observations of a previously observed landmark are ignored. With optimization only, all observations are used to impose soft constraints on the final map, but information from other SLAM-performing robots is not used. Optimization and sharing fully enables CG-SLAM as described in earlier sections.

Resulting maps as obtained by robot number 1 are shown in Figures 2-4 alongside the ground-truth map in Figure 5. We also include tables of landmark pose errors averaged across all landmarks and pose errors averaged across all timesteps in Tables I and II; the best achieved result for each individual robot, as well as the best achieved average across all robots, are highlighted in bold. As all robots in Dataset 2 eventually locate all landmarks whether maps are shared or not, the average landmark error metric will not be distorted by any missing landmarks. (To avoid overlap in error measurements, each robot only measures robot pose error for its own poses, even if its map includes nodes representing other robots.)

We observe from Tables I and II that for every number of robots active in the simulation, optimization consistently reduces both average landmark pose error when averaged across all robots, and that enabling map sharing consistently reduces both types of error further. The reduction in error is not as large for robot pose error, yet the pattern remains that pose error averaged across robots is reduced by optimization and reduced further by map sharing, the two-robot case notwithstanding.

However, for a given number of robots, sharing does not always reduce an individual robot's average pose error; for instance, in Table I, when simulating with two active robots we see that sharing actually *increases* Robot 1's average landmark error from 0.755 meters to 1.034 meters. A possible explanation for this is that graph optimization with map-sharing attempts to find a map that minimizes inconsistency between robots' observations of the same landmarks, meaning that if the two robots misplace a particular landmark in roughly the same direction (e.g. underestimating its x-coordinate by 0.5 and 1, respectively), the optimized map will misplace the landmark by an amount in between the two incorrect estimates, reducing the worse robot's error but increasing the better robot's. This issue is mitigated with the addition of more robots, as exchanging maps with more robots observing the same landmarks from different positions reduces the likelihood of a landmark being consistently misplaced in a similar direction by all observers.

Indeed, the benefit of sharing maps between robots appears to improve with more robots, as for average pose error across all robots, as well as for every individual robot except Robot 2, the lowest average landmark and robot pose errors are consistently achieved when optimizing and sharing maps between five robots, the maximum possible number of robots in the simulation.

TABLE I

LANDMARK POSE ERROR WITH INITIAL ROBOT POSES KNOWN

		Avg. L_2 landmark pose error (m)		
# Robots	Robot ID	No opt./sharing	opt. only	opt. + sharing
1	1	5.015	0.755	-
2	1	5.015	0.755	1.034
	2	3.647	1.360	0.900
	(Average)	4.331	1.007	0.967
3	1	5.015	0.755	0.593
	2	3.647	1.360	**0.558**
	3	2.541	0.375	0.471
	(Average)	3.736	0.830	0.541
4	1	5.015	0.755	0.471
	2	3.647	1.360	0.691
	3	2.541	0.375	0.601
	4	2.810	1.703	0.710
	(Average)	3.505	1.048	0.674
5	1	5.015	0.755	**0.403**
	2	3.647	1.360	0.667
	3	2.541	0.375	**0.358**
	4	2.810	1.703	**0.361**
	5	0.808	0.619	**0.374**
	(Average)	2.966	0.962	**0.433**

In general, we show that optimizing the pose-graph maps generated from robot measurements reduces pose error across the graph, and that enabling robots to share their maps reduces it further. Furthermore, the improvement appears to increase with a larger number of robots.

D. Extensions and Improvements

While we demonstrated that introducing map-sharing between robots performing graph-based SLAM can reduce pose error across the robots, in absolute terms our best observed robot pose error of 0.882 meters per timestep is still fairly high. This is partially a consequence of our choice

TABLE II

ROBOT POSE ERROR WITH INITIAL ROBOT POSES KNOWN

		Avg. L_2 robot pose error (m)		
# Robots	Robot ID	No opt./sharing	opt. only	opt. + sharing
1	1	5.982	1.139	-
2	1	5.015	1.139	1.622
	2	3.147	1.481	1.029
	(Average)	4.565	1.310	1.325
3	1	5.982	1.139	1.235
	2	3.147	1.481	**0.773**
	3	5.372	0.783	0.820
	(Average)	4.834	1.134	0.943
4	1	5.982	1.139	1.250
	2	3.147	1.481	0.816
	3	5.372	0.783	0.836
	4	3.362	1.285	0.986
	(Average)	4.466	1.172	0.972
5	1	5.982	1.139	**1.112**
	2	3.147	1.481	0.828
	3	5.372	0.783	**0.771**
	4	3.362	1.285	**0.915**
	5	7.428	0.918	**0.783**
	(Average)	5.058	1.121	**0.882**

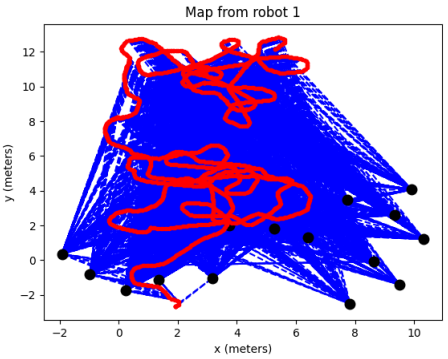

Fig. 2. The final map obtained by robot 1 of the setup from Dataset 2 with no sharing or optimization. Robots were initialized with ground-truth poses and orientations, but used odometry readings for state estimation thereafter. The robot's estimated poses are depicted in red, estimated landmark poses are depicted in black, and graph edges (sensory observations) are depicted in blue.

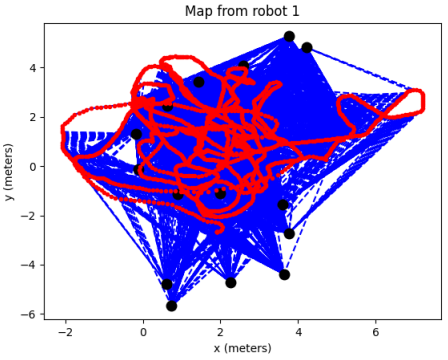

Fig. 3. The final map obtained by robot 1 of the setup from Dataset 2, optimizing maps but not sharing with others. Robots were initialized with ground-truth poses and orientations, but used odometry readings for state estimation thereafter.

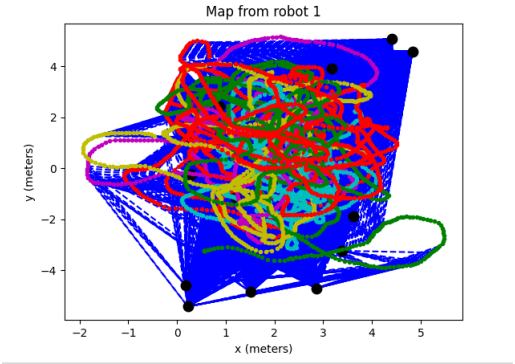

Fig. 4. The final map obtained by robot 1 of the setup from Dataset 2, including the merged map after receiving observations and estimated poses from other robots. Robots were initialized with ground-truth poses and orientations, but used odometry readings for state estimation thereafter. Robot poses are color-coded according to the robot they were recorded by.

Fig. 5. The ground-truth map of robot 1's motion and landmark observations.

of optimization interval of 250 seconds during experimental trials, as well as potentially having chosen a sub-optimal edge information matrix, but also suggests that further improvement upon our approach should be possible.

One aspect that may be improved is the map-merging step. As robots must align on a common global coordinate frame to share maps, and robots cannot visually identify each other's orientation, CG-SLAM requires robots to first reach a consensus on their relative position, which is only possible when the robots can observe each other simultaneously, an occurrence which is not frequent. One approach to make map sharing less restrictive is to reach consensus on *landmark* poses instead of robot poses. This can be done through Iterated Closest Point (ICP) to align the set of landmarks both robots have observed prior to meeting, yielding a transformation one robot can apply to its map to match the orientation and frame of the other.

As ICP landmark alignment would not require the robots to observe each other, robots could exchange maps *any* time they are within communication range. Additionally, that after aligning their frames, a robot could use the self-reported orientation of a robot it observes to complete the $\mathcal{SE}(2)$ pose estimation of that other robot, no longer requiring us to discard observations of other robots. A potential drawback is that, as the number of landmarks in these datasets is small and an individual robot's landmark error may be high, the optimal alignment yielded by ICP may be incorrect, introducing more error to every observation in the map after applying the transformation.

Additionally, CG-SLAM optimizes maps in regular intervals, the choice of which is user-defined and subsequently quite arbitrary. Loop closures in SLAM often re-optimize the map every time a revisited location is identified; in our context, this would likely be when a previously seen landmark is re-observed. However, this would make optimization a much more common operation than it was in our experimental trials, potentially making the simulation very time-consuming to run and reducing the feasibility of CG-SLAM being used in realistic real-time applications.

Visual Bag-of-Words (BoW) methods are often used for this purpose, recognizing previously visited locations by identifying features from the camera observations. ORB is a strong candidate for feature detection and description owing to its balance between speed and performance, while FAB-MAP, a system for appearance-only navigation and scene recognition, is a likely choice of scene recognition algorithm. [11] Furthermore, once a place is recognized, the eight-point algorithm can be used to identify the relative pose between the two observations of the scene. [12] This would allow us to impose additional constraints on the pose graph, further reducing error of the optimized map. As MR.CLAM Dataset 7 includes the camera images from each robot that were used to obtain the dataset's range and bearing measurements, that dataset would be highly convenient for testing CG-SLAM with visual scene recognition against our earlier forms of CG-SLAM.

V. CONCLUSION

We present a cooperative graph-based SLAM method to reduce estimated pose error in multi-robot SLAM by exchanging maps between robots. We define methods to estimate robot and landmark state from visual and odometry measurements and describe a process to optimize the pose graph constructed from these measurements, as well as a process to merge maps between different robots. The resulting algorithm is evaluated comparing accuracy with and without graph optimization and map sharing in a simulated environment using a pre-recorded dataset. We conclude that sharing maps between robots is an effective way to reduce error in graph-based SLAM, and that the benefits increase as more robots participate in the mapping.

ACKNOWLEDGMENT

The UTIAS MR.CLAM datasets [10] were provided with MATLAB scripts to load the datasets and sample from them to ensure uniformly spaced measurement timesteps. These were rewritten in Python prior to beginning simulation. A script to animate the ground truth of each dataset was also provided, which was used to generate Figure 1.

REFERENCES

[1] S. Thrun, Simultaneous localization and mapping, in Robotics and Cognitive Approaches to Spatial Mapping, Berlin, Germany: Springer, 2007, pp. 13-41.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, FastSLAM: A factored solution to the simultaneous localization and mapping problem, in Proc. Ntl. Conf. on Artificial Intelligence, Edmonton, Canada, 2002, pp. 593-598

[3] 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.

[4] J. G. Morrison, D. Gálvez-López, and G. Sibley, MOARSLAM: Multiple operator augmented RSLAM, in Distributed Autonomous Robotics Systems, Tokyo, Japan: Springer, 2016, pp. 119-132

[5] A. Gil, Ó. Reinoso, M. Ballesta, and M. Juliá, Multi-robot visual SLAM using a Rao-Blackwellized particle filter, Robotics and Autonomous Systems, vol. 58, no. 1, pp. 68-80, 2010.

[6] V. S. Menon, Decentralized approach to SLAM using computationally limited robots, M.S. thesis, Worcester Polytechnic Inst., Worcester, MA, May 2017.

[7] I. Deutsch, M. Liu, and R. Siegwart, A framework for multi-robot pose graph SLAM, in 2016 Proc. Int. Conf. on Real-time Computing and Robotics, pp. 567-572.

[8] S. Nagavalli, A. Lybarger, L. Luo, N. Chakraborty, and K. Sycara, Aligning coordinate frames in multi-robot systems with relative sensing information, in 2014 Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp.388-395.

[9] K. Y. Leung, T.D. Barfoot, and H. H. Liu, Decentralized localization of sparsely-communicating robot networks: a centralized-equivalent approach, IEEE Transactions on Robotics, vol. 26, no. 1, pp. 62-77, 2009.

[10] K. Y. Leung, Y. Halpern, T. D. Barfoot, and H. H. Liu, The UTIAS multi-robot cooperative localization and mapping dataset, Intl. Journal of Robotics Research, vol. 30, no. 8, pp. 969-974.

[11] M.J. Cummins, and P. M. Newman, FAB-MAP: appearance-based place recognition and mapping using a learned visual vocabulary model, in Proc. 27th Intl. Conf. on Machine Learning, pp. 3-10.

[12] H. C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, in Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, M.A. Fischler and O. Firschein, Eds., 1987, pp. 61-62.