2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Overlapping Domain Decomposition for Distributed Pose Graph Optimization

Aneesa Sonawalla[1,2], Yulun Tian[3], and Jonathan P. How[1]

***Abstract*— We present ROBO (Riemannian Overlapping Block Optimization), a distributed and parallel approach to multi-robot pose graph optimization (PGO) based on the idea of overlapping domain decomposition. ROBO offers a middle ground between centralized and fully distributed solvers, where the amount of pose information shared between robots at each optimization iteration can be set according to the available communication resources. Sharing additional pose information between neighboring robots effectively creates *overlapping* optimization blocks in the underlying pose graph, which substantially reduces the number of iterations required to converge. Through extensive experiments on benchmark PGO datasets, we demonstrate the applicability and feasibility of ROBO in different initialization scenarios, using various cost functions, and under different communication regimes. We also analyze the tradeoff between the increased communication and local computation required by ROBO's overlapping blocks and the resulting faster convergence. We show that overlaps with an average inter-robot data cost of only 36 Kb per iteration can converge 3.1× faster in terms of iterations than state-of-the-art distributed PGO approaches. Furthermore, we develop an asynchronous variant of ROBO that is robust to network delays and suitable for real-world robotic applications.**

I. Introduction

In GPS-denied scenarios, autonomous operation of robot teams relies on robust and accurate trajectory estimation from noisy relative measurements. *Pose graph optimization* (PGO) forms the backbone of GPS-denied, long-term localization in state-of-the-art multi-agent systems [1]–[3]. Many systems adopt a centralized architecture [4]–[6], in which a central computation node receives measurements from all robots and solves the optimization using off-the-shelf, high-performance libraries such as GTSAM [7] and Ceres Solver [8]. To handle increasing scale, inter-robot privacy constraints, and real-world network connectivity, an alternative line of work takes a distributed approach to PGO [2], [9], [10]. There are several state-of-the-art distributed PGO (DPGO) methods that offer minimal inter-agent communication per iteration, resilience to network delays, limited trajectory sharing between robots, and even certifiable optimality [11]–[14]. Yet these methods suffer from poor convergence rates, often requiring hundreds or thousands of iterations to reach a high-precision solution that a centralized solver could reach in tens of iterations. The limited inter-robot communication of these methods is appropriate for robotics applications using low-bandwidth wireless networks, such as the 0.25 Mbps achievable with the Digi XBee 3 Zigbee 3.0 module [15]. However, modern ground and air robots frequently carry modules with much higher throughput, like the 100 Mbps Silvus SL5200 radios [16]. In these cases, existing DPGO methods significantly under-utilize the available communication bandwidth. In this work, we address this gap by investigating a middle ground design between the fully centralized and fully distributed extremes that can flexibly take advantage of the resources available. Our method achieves substantially improved convergence rates via a controlled increase in communication at every optimization iteration, maintaining communication requirements well within the limits of modern networks.

To this end, we present **ROBO** (Riemannian Overlapping Block Optimization), a novel, parallel, and fast distributed PGO solver based on overlapping domain decomposition, a classical approach in numerical optimization [17]. We design an overlapping scheme for solving graph-structured optimization problems with on-manifold decision variables, drawing on prior works [18]–[20] that have investigated similar decomposition approaches. Multi-robot PGO naturally decomposes the larger PGO problem into blocks, or domains, corresponding to individual robots. Existing DPGO methods only require agents to communicate poses located at the boundary of their respective domains at every iteration. With ROBO, robots selectively communicate additional information from the interiors of their domains. This allows the original domains to inflate and overlap with each other, providing each robot more information on the global problem to accelerate convergence. The degree of overlap, controlled by a hyperparameter ω, allows our approach to trade off per-iteration communication and overall convergence rate. Without overlap ($\omega = 0$), our approach reduces to the previous distributed solver [12] that operates on disjoint domains. Meanwhile, with increasing overlap ($\omega \to \infty$), ROBO yields substantially faster convergence, eventually recovering centralized computation where every robot solves the global pose graph. In practice, the degree of overlap applied can be set according to the available network bandwidth, allowing our method to be adapted to available resources to achieve faster convergence. Further, the parallel nature of ROBO makes the method readily applicable under asynchronous communication. Finally, while it might be expected that sharing more information improves estimation accuracy, ROBO's flexible design allows us to ask: *how much* overlap is needed to improve convergence in common DPGO settings, and

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 2141064, the Draper Scholars Program, and in part by ARL DCIST under Cooperative Agreement Number W911NF-172-0181.

[1]Aneesa Sonawalla and Jonathan P. How are with the MIT Department of Aeronautics and Astronautics. {aneesa, jhow}@mit.edu.

[2]Draper Scholar, The Charles Stark Draper Laboratory, Cambridge, MA.

[3]Yulun Tian is with the University of Michigan Robotics Department. yulunt@umich.edu.

is the increased computation/communication tractable for robotics applications? To the authors' knowledge, this is the first demonstration of convergence of this type of overlapping domain decomposition method for distributed, on-manifold optimization, as well as the first communication cost analysis for this type of approach. In summary, our contributions are:

- **ROBO** (Riemannian Overlapping Block Optimization), a *fully parallel* PGO method that leverages overlapping domain decomposition to converge in substantially fewer iterations than state-of-the-art approaches;
- An analysis of the convergence improvement and communication cost trade-off when increasing overlap;
- An asynchronous variant of ROBO that is robust to network delays common in real-world applications;
- Extensive experiments on both synthetic and real-world benchmark datasets demonstrating ROBO's adaptability to different initialization schemes, optimization formulations, and communication paradigms.

II. Related Work

A. Multi-Agent PGO

Multi-agent PGO involves solving a nonlinear, non-convex optimization for multiple robots' poses given a set of noisy relative inter- and intra-robot measurements. This optimization can be solved in a centralized, decentralized, or distributed fashion. Fast, mature solvers exist for centralized multi-robot PGO [21]–[23], but centralized computation can become untenable as the size of the robot team or operational area scales up. Decentralized methods [24]–[27] remove the need for a central server, but can suffer under communication or local computation constraints. This work focuses on distributed methods, which enable highly scalable multi-robot localization under real-world constraints [28].

B. Distributed PGO

Distributed PGO (DPGO) has been a problem of interest for many years, particularly over the past decade. Choudhary et al. introduced a two-stage approach in DGS that uses the successive over-relaxation and the Jacobi over-relaxation [9]. Tian et al. developed DC2-PGO, which uses a distributed Riemannian staircase approach to solve a semidefinite relaxation of the PGO problem and can certify global optimality of the returned solution [12]. The authors also introduced Riemannian block coordinate descent (RBCD) and its accelerated variant RBCD++, which have provable convergence on any smooth optimization defined on the product of Riemannian manifolds [12]. Fan and Murphey introduced a provably convergent method that leverages an accelerated majorization minimization approach, along with an accelerated, distributed initialization scheme for multi-agent PGO [13], [29]. Recently, Li et al. [30] improved the acceleration of RBCD/RBCD++ with IRBCD and further proposed a load-balanced graph partitioning approach in constructing the distributed optimization.

All of the above methods require synchronization across agents; other DPGO approaches exist that do not. Cunningham et al. introduced DDF-SAM, a decentralized system in which agents share reduced versions of their local graphs obtained from Gaussian elimination [26], [27]. Tian et al. developed ASAPP, a distributed and asynchronous PGO approach that remains provably convergent under bounded network delay [11]. Murai et al. proposed a method based on Gaussian belief propagation that is fully asynchronous, distributed, and robust to communication delays and dropouts [10]. McGann et al. took an approach based on consensus alternating direction method of multipliers (C-ADMM) to develop MESA, which also empirically converges under communication limitations [14].

While the above methods have all made significant advances in DPGO in terms of convergence rate, robustness, and guaranteed convergence, the current state-of-the-art still requires many iterations to reach a high-precision solution that a centralized solver could reach in tens of iterations or fewer. These methods most often assume a maximally restrictive communication scheme in the interest of privacy constraints and limited network bandwidth, meaning they cannot take advantage of higher-throughput networks increasingly available in modern robotics applications.

C. Overlapping Domain Decomposition

Overlapping domain decomposition, also known as the overlapping Schwarz method, has historically been applied to solve large, sparse linear systems as a "divide-and-conquer" strategy [17]. Shin et al. proved the convergence in both synchronous and asynchronous settings of an overlapping domain decomposition approach to graph-structured optimization problems that can be expressed as an unconstrained, convex quadratic program [18]. A subsequent work extended this result to constrained quadratic programs in [19]. The basis for the authors' convergence proofs is that the sensitivity of the solution at any node in the graph to a change in the solution at another node decays exponentially with the distance between those nodes [20]. A similar property was shown in [31] for optimization problems with locally coupled costs. Shin et al. showed this sensitivity property holds for graph-structured nonlinear programs whose objective functions meet certain conditions [20], while Na et al. showed it holds for nonlinear optimal control problems [32]. While these works developed Schwarz-like schemes for a variety of graph-structured optimization problems, to the best of our knowledge, the literature does not include an exploration of convergence for the case of graph-structured optimization with on-manifold decision variables like PGO. Additionally, there is no analysis of the tradeoff between communication cost and convergence rate that is essential to consider in real-world robotics applications.

III. Problem Formulation

In this section, we formally introduce the distributed PGO problem and relevant mathematical preliminaries. Our formulation is compatible with multiple commonly used objective function definitions. We also consider several communication scenarios relevant to real-world DPGO applications.

A. Preliminaries

The *special orthogonal group* is denoted as $\mathrm{SO}(d) \triangleq \{R \in \mathbb{R}^{d\times d} \,|\, R^\top R = I_d, \det(R) = 1\}$, where I_d is the d-dimensional identity matrix and $d = 2$ or $d = 3$ for 2D or 3D poses. The *special Euclidean group* is denoted by $\mathrm{SE}(d) \triangleq \mathrm{SO}(d) \ltimes \mathbb{R}^d$. The set of n poses to be estimated are $T_1, \ldots, T_n$, where $T_i = (R_i, t_i)$ includes rotation $R_i \in \mathrm{SO}(d)$ and translation $t_i \in \mathbb{R}^d$. We consider noisy, full-rank relative measurements between poses also consisting of a rotation and translation, with the measurement between poses i and j given by $\widetilde{T}_{ij} = (\widetilde{R}_{ij}, \widetilde{t}_{ij})$, $\widetilde{R}_{ij} \in \mathrm{SO}(d)$ and $\widetilde{t}_{ij} \in \mathbb{R}^d$.

B. Pose Graph Optimization

We can model the PGO problem as a directed, connected graph $G = (V, E)$ in which the set of nodes V corresponds to pose variables $\{T_i\}_i^{|V|}$ to be estimated, and the set of edges E correspond to pairwise relative measurements $\{\widetilde{T}_{ij}\}$ from i to j. $|V|$ is the size of the set V, equivalent to the number of poses to be estimated. The PGO problem is formulated as

$$\begin{aligned} &\min_{\{T_i\}_{i\in V}} \frac{1}{2} \sum_{(i,j)\in E} \mathrm{dist}^2_{\Omega_{ij}}(T_i\widetilde{T}_{ij}, T_j), \\ &\text{s.t.} \quad R_i \in \mathrm{SO}(d),\ t_i \in \mathbb{R}^d \quad \forall\, T_i = (R_i, t_i), i \in V. \end{aligned} \tag{1}$$

In (1), $\mathrm{dist}^2_{\Omega_{ij}} : \mathrm{SE}(d) \times \mathrm{SE}(d) \to \mathbb{R}_{\geq 0}$ denotes the squared distance between two poses weighted by an information matrix Ω_{ij}. Following [11], [12], [33], we assume the translation and rotation noise are independent and isotropic. This means that each Ω_{ij} can be fully described by two scalars $\kappa_{ij}, \tau_{ij} > 0$ corresponding to the precisions of the rotation and translation measurements, respectively. Using the chordal distance metric for rotations, this formulation recovers the same objective function used by SE-Sync [33] and DC2-PGO [12], where $\mathrm{dist}^2_{\Omega_{ij}}(T_i\widetilde{T}_{ij}, T_j) = \kappa_{ij}||R_j - R_i\widetilde{R}_{ij}||^2_F + \tau_{ij}||t_j - t_i - R_i\widetilde{t}_{ij}||^2_2$. We also consider an alternative objective that uses the geodesic distance for rotations. In this case, $\mathrm{dist}^2_{\Omega_{ij}}(T_i\widetilde{T}_{ij}, T_j) = \kappa_{ij}||\,\mathrm{Log}(\widetilde{R}^T_{ij}R^T_i R_j)||^2_2 + \tau_{ij}||t_j - t_i - R_i\widetilde{t}_{ij}||^2_2$. The logarithm map $\mathrm{Log}(\cdot)$ converts a rotation matrix to its angle-axis representation [34].

For DPGO, a team of N robots collaboratively solves (1) without a centralized node. Each robot solves a subset of poses in V. We consider each robot's trajectory, or its "owned" poses, as a subgraph, providing a natural partitioning of $G(V, E)$ into *blocks*, which we denote by V_α with $\alpha \in \mathcal{B} \triangleq \{1, 2, \ldots, N\}$. Blocks are connected by inter-robot measurements to form the global, connected graph. Figure 1 shows an example of a three-robot graph with intra- and inter-robot measurements with different colors (blue, orange, green) highlighting V_α, V_β, V_γ.

IV. Methods

In this section, we start by reviewing Riemannian block coordinate descent (RBCD) [12] and describe how we extend RBCD to develop ROBO. We then detail how our method is adapted for the synchronous, edgewise, and asynchronous communication settings drawn from prior works [11]–[14].

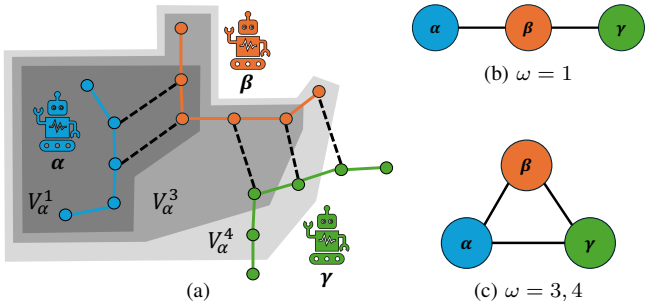


Fig. 1. Example three-robot pose and communication graphs. (a) Nodes correspond to poses; node color indicates "ownership" of that pose. Solid lines denote odometry measurements, dashed lines are inter-robot loop closures. Overlapping blocks with $\omega = 1, 3, 4$ highlighted for robot α. Robot subgraphs summarized in (b) and (c) to form the communication graph, $G^\omega_C(\mathcal{B}, \mathcal{E})$. (b) $G^\omega_C(\mathcal{B}, \mathcal{E})$ for $\omega = 1$. (c) $\omega = 3$ or 4 changes $G^\omega_C(\mathcal{B}, \mathcal{E})$ due to inter-robot measurements, such that all robots are connected.

A. ROBO: PGO with Overlapping Domain Decomposition

Block coordinate descent (BCD) methods are well-suited to DPGO because of the natural decomposition of the global pose graph into blocks corresponding to individual robots. RBCD was introduced in [12] as a general BCD algorithm applicable to smooth optimization problems over the product of Riemannian manifolds. This class of optimization problems includes the PGO problem in (1). To collaboratively perform PGO using RBCD, each robot only needs to share the "boundary" poses from its own block that are connected to neighboring robots by an inter-robot measurement. From a centralized view, the basic steps of an RBCD iteration are 1) select a block (i.e., robot) to optimize; 2) approximately solve the optimization over that block, holding all other nodes in the graph constant; and 3) carry over all other values in the global graph.

ROBO improves convergence rates by allowing more per-iteration communication than RBCD. This is most useful when the primary goal is rapid convergence and resources afford higher inter-robot communication bandwidth, which is typical in many real-world settings with ground or air robots. With ROBO, we extend the idea of RBCD to include *overlapping* blocks constructed from overlapping domain decomposition [17]. Blocks no longer correspond just to one robot's poses (see Figure 1). We further choose to optimize over all blocks in parallel, rather than alternating as in [12].

We first show how to create these overlapping blocks from the global graph. Following [18], we define the overlapping decomposition using the graph distance: let $d_G(i, j)$ be the number of edges in the shortest path between nodes $i, j \in V$. The distance between a node $i \notin X$ and a set of nodes X is $d_G(i, X) = \min_{j\in X} d_G(i, j)$. If $i \in X$, then $d_G(i, X) = 0$. The overlapping set of nodes for robot α with overlap size ω is then defined as $V^\omega_\alpha \triangleq \{i \in V \mid d_G(i, V_\alpha) \leq \omega\}$. Note that V^ω_α also includes robot α's nodes, i.e., $V_\alpha \subseteq V^\omega_\alpha$.

In ROBO, each robot $\alpha \in \mathcal{B}$ solves a subproblem of (1) over its block V^ω_α. We denote the edges within the block V^ω_α, or the "interior" edges of V^ω_α, by $E^\omega_\alpha \triangleq \{(i, j) \in E \mid i, j \in V^\omega_\alpha\}$. Note that E^ω_α includes all edges—odometry and all loop closures—satisfying this definition. The block V^ω_α has boundary poses $V^{\omega+1}_\alpha \setminus V^\omega_\alpha$. When the remaining poses in

Algorithm 1 ROBO: Overlapped Distributed PGO

Input: $G(V, E)$, ω, $\{V_\alpha\}$, convergence condition
Output: V at a local minimum of (1)
1: Determine $G^\omega_C(\mathcal{B}, \mathcal{E})$ using $G(V, E)$, $\{V_\alpha\}$
2: Share initial estimates to form $\{V_\alpha^{\omega+1}\}$ $\forall$ agents $\alpha \in \mathcal{B}$
3: **while** not converged **do**
4: **for** $\alpha \in \mathcal{B}_{\text{conn}}$ (in parallel) **do**
5: $V_\alpha^\omega \leftarrow$ approximately solve (2)
6: Send $\{v_\alpha \in V_\alpha \mid v_\alpha \in V_\beta^{\omega+1}\}$ to $\beta \in \mathcal{B}_{\text{conn}},\ (\alpha, \beta) \in \mathcal{E}$
7: $V_\alpha^{\omega+1} \backslash V_\alpha \leftarrow$ updates from $\beta \in \mathcal{B}_{\text{conn}},\ (\alpha, \beta) \in \mathcal{E}$
8: **end for**
9: **end while**

$V \backslash V_\alpha^\omega$ are fixed, these new boundary poses effectively form *prior terms* that constrain the robot's local subproblem. Let $\bar{T}_i$ denote a fixed boundary pose i. Since G is directed, the edges connecting the fixed poses to the block V_α^ω can be either "incoming" or "outgoing" with respect to V_α^ω. Let the outgoing edges be $E_{\alpha,out}^{\omega+1\backslash\omega} \triangleq \{(i,j) \in E \mid i \in V_\alpha^\omega,\ j \in V_\alpha^{\omega+1} \backslash V_\alpha^\omega\}$, and the incoming edges be $E_{\alpha,in}^{\omega+1\backslash\omega} \triangleq \{(i,j) \in E \mid i \in V_\alpha^{\omega+1} \setminus V_\alpha^\omega,\ j \in V_\alpha^\omega\}$. With these definitions, the subproblem solved by robot α is:

$$\begin{aligned} \min_{\{T_i\}_{i \in V_\alpha^\omega}} \ & \frac{1}{2} \sum_{(i,j) \in E_\alpha^\omega} \text{dist}^2_{\Omega_{ij}}(T_i \widetilde{T}_{ij}, T_j) \\ & + \frac{1}{2} \sum_{(i,j) \in E_{\alpha,out}^{\omega+1\backslash\omega}} \text{dist}^2_{\Omega_{ij}}(T_i \widetilde{T}_{ij}, \bar{T}_j) \\ & + \frac{1}{2} \sum_{(i,j) \in E_{\alpha,in}^{\omega+1\backslash\omega}} \text{dist}^2_{\Omega_{ij}}(\bar{T}_i \widetilde{T}_{ij}, T_j) \\ \text{s.t.} \quad & R_i \in \text{SO}(d),\ t_i \in \mathbb{R}^d \quad \forall\, T_i = (R_i, t_i) \in V_\alpha^\omega. \end{aligned} \tag{2}$$

Remark (Communication graph)**.** *As in RBCD [12], the subproblem blocks induce a robot-level communication graph. To form the local subproblem, robot α needs V_α^ω and E_α^ω; the boundary poses, $V_\alpha^{\omega+1} \backslash V_\alpha^\omega$; and the incoming and outgoing boundary edges, $E_{\alpha,in}^{\omega+1\backslash\omega}$ and $E_{\alpha,out}^{\omega+1\backslash\omega}$. This means robot α needs to form $V_\alpha^{\omega+1}$ to solve the subproblem for overlap ω. If we summarize $V_\alpha^{\omega+1}$ as a single node, we get a second graph showing the robot-level level connectivity needed for overlap ω. We call this graph the* communication graph*, denoted $G^\omega_C(\mathcal{B}, \mathcal{E})$. An edge exists in $\mathcal{E}$ between $\alpha \in \mathcal{B}$ and $\beta \in \mathcal{B}$ if and only if there is at least one pose $i \in V_\alpha^{\omega+1}$ such that $i \in V_\beta$ or at least one pose $j \in V_\beta^{\omega+1}$ such that $j \in V_\alpha$.*

Figure 1 illustrates the above concepts. Figure 1a shows V_α^ω for robot α in shades of gray for $w = 1, 3, 4$. If we choose $\omega = 3$, the three poses in $V_\alpha^4 \setminus V_\alpha^3$ are also needed by robot α to form the optimization in (2). Figures 1b and 1c show the communication graphs for several values of ω where the example pose graph from Figure 1a leads to different communication connectivity, depending on ω.

We can now detail the steps of our method (Algorithm 1). We assume the full graph $G(V, E)$, the overlap size ω, and the graph partitioning to create $\{V_\alpha\}$ are known. The measurement edges E and the partitioning V_α are sufficient to determine the overlapped blocks, $\{V_\alpha^\omega\}$, and the blocks with boundary edges and poses included, $\{V_\alpha^{\omega+1}\}$ for $\alpha \in \mathcal{B}$. The communication graph $G^\omega_C(\mathcal{B}, \mathcal{E})$ can then be determined using $\{V_\alpha^{\omega+1}\}$. Different initialization schemes can be used in Step 2; several options are explored in Section V.

In each iteration of ROBO, agents solve their inflated subproblems in parallel, then project their local solutions onto their own domains $\{V_\alpha\}$ before sharing the required parts of their solutions with their neighbors. This is reflected in Steps 4–8. In general, we can have a different subset of robots $\mathcal{B}_{\text{conn}} \subseteq \mathcal{B}$ that perform local optimization at every iteration. In Step 5, each agent in $\mathcal{B}_{\text{conn}}$ approximately solves the subproblem in (2) locally over its block V_α^ω, where $V_\alpha^{\omega+1} \backslash V_\alpha^\omega$ are used to construct the priors in (2). Here we use a single step of a Levenberg-Marquardt solver initialized with V_α^ω from the last iteration to reach an approximate solution to (2). However, the method is not specific to which local solver is used. In Step 6, each agent in $\mathcal{B}_{\text{conn}}$ shares the subset of its owned poses V_α that are also in its neighbors' overlapped blocks with each of those neighbors; that is, each agent α sends poses $v_\alpha \in V_\alpha$ to its neighbor β if $v_\alpha \in V_\beta^{\omega+1} \setminus V_\beta$. We emphasize that only the overlapped portions of each agent's own and neighboring blocks are shared, not the entire set of poses V_α. Finally, in Step 7, each agent in $\mathcal{B}_{\text{conn}}$ updates its local graph $V_\alpha^{\omega+1}$ with the overlapped poses $V_\alpha^{\omega+1} \backslash V_\alpha$ received in Step 6. The convergence condition may be set by local convergence criteria, by periodic synchronization across robots to determine global properties, or by a practical need such as a maximum allowable optimization time.

B. Communication Schemes

By default, the approach in Algorithm 1 is synchronized in that all agents in $\mathcal{B}_{\text{conn}}$ must complete one round of parallel optimization and communication before proceeding to the next iteration. In the following, we discuss how Algorithm 1 can be executed in different communication scenarios.

Synchronous ROBO. In the *synchronous* communication paradigm, all agents are able to communicate with their neighbors at each iteration, and iterations are synchronized across the team. Thus, in Step 4, $\mathcal{B}_{\text{conn}} = \mathcal{B}$. This paradigm leads to the most straightforward algorithm design, but it also places the most stringent requirements on the physical network, and is therefore best suited to applications with reliable network connections.

Edgewise ROBO. As operating environment and team size scale, it is likely that not all agents will be in constant communication, meaning $\mathcal{B}_{\text{conn}}$ in Step 4 will be time-varying and in general a subset of $\mathcal{B}$. To model this limited network connectivity, we consider an *edgewise-only* scheme, inspired by [14]. We assess the performance of our method in the most restrictive edgewise case as in [14], wherein only two agents can communicate at every iteration. In this regime, an agent may optimize using stale data from agents it has not communicated with recently. We simulate this by randomly sampling a robot pair $(\alpha, \beta) \in \mathcal{E}$ at every iteration and setting $\mathcal{B}_{\text{conn}} = \{\alpha, \beta\}$ for that iteration. The rest of Algorithm 1 remains the same.

Asynchronous ROBO. Lastly, we consider the *asynchronous* regime. While the edgewise scenario relaxes communication requirements across the team, it still assumes alternating communication and optimization steps. In contrast, the asynchronous regime (inspired by [11]) fully decouples the communication and optimization processes. At any given time, an agent may optimize locally with the latest information it has. We assume that all agents will eventually be able to connect to other agents over the network, possibly after some amount of delay due to network latency or dropout. Thus, any robot may be performing local optimization with stale data from its neighbors. This paradigm mimics a real-world implementation with delays.

We follow [11] and implement asynchronous ROBO by separating the communication and optimization tasks into parallel software processes. When connected to neighboring robots, the communication process sends the overlapped portions of its latest local estimate V_α to those robots (Step 6) and handles incoming updates. This process receives and caches neighbor state updates for the optimization process to access in Step 7. The optimization process meanwhile operates in a loop in which it 1) updates the local state, $V_\alpha^{\omega+1}$using the cached updates from the communication process; 2) solves (2) over V_α^ω; and 3) provides the latest V_α back to the communication process. As in [11], we model the individual robots' choice of when to optimize as a Poisson process with rate $\lambda > 0$, i.e., the times $t_k \geq 0$ between successive optimizations follow an exponential distribution, $P(t_k \leq a) = 1 - e^{-\lambda a}$ for $a \geq 0$. All agents share the same λ, which is set according to the robots' local computation capacity and the network capacity. Note that both processes must access the shared cache containing the latest local estimate for V_α and the latest neighbor states. Each access must be done atomically to maintain data integrity. In practice, this can be easily implemented using software locks on the cache.

V. Experiments

We examine the performance of ROBO with different levels of overlap under the three communication paradigms described in Section IV. We use chordal distance as our distance metric unless explicitly noted. For all experiments below, we use a collection of 6 simulated and 16 real-world benchmark PGO datasets[1,2]. This collection consists of 15 datasets with 2D measurements and 7 with 3D measurements. For the `garage` dataset, we use noise standard deviations of 0.4 deg and 0.04 m to avoid numerical instabilities caused by the unrealistically large uncertainties (> 40 deg) in the original dataset. For each benchmark, we use SE-Sync [33] to compute a reference solution and optimal cost for the centralized PGO problem. When using the geodesic distance metric, we find a reference solution by solving the centralized problem using Ceres [8] starting from chordal initialization. We show results only up to $\omega = 3$ as we observe diminishing returns in convergence improvement for $\omega > 3$.

[1]https://github.com/mit-acl/dpgo/tree/main/data

[2]https://github.com/MurpheyLab/DPGO/tree/main/dataset

The benchmarks used are single-agent datasets, so we partition them to simulate multiple robots by segmenting the datasets sequentially into subgraphs according to the pose numbering provided within each dataset. We simulate 5 robots for all benchmarks. When available, we compare our method against state-of-the-art baseline methods that use the same communication paradigm and distance metric. For the methods we compare against, we use the implementations provided by the authors, unmodified except where noted.

A. Evaluation Metrics

We consider two platform-agnostic metrics: RMSE absolute position error (APE) [35] and *relative suboptimality*. We define relative suboptimality at iteration k as $\Delta_{rel} = (F_k - F^*)/F^*$. F_k is the value of the cost in (1) evaluated at iteration k, and F^* is the optimal cost for the given graph. To evaluate F_k in a distributed setting, we aggregate the agents' solutions using natural ownership of each pose to compute a centralized F_k at each iteration; e.g., we use robot α's solutions for all of robot α's poses, even if robot β is locally estimating part of robot α's trajectory. We use the same aggregated solution to compute APE, using the optimal solution found with either SE-Sync or Ceres (depending on distance metric used) as the reference trajectory.

B. Synchronous ROBO

Synchronous ROBO follows Algorithm 1 with $\mathcal{B}_{\text{conn}} = \mathcal{B}$. ROBO-$\omega$ denotes ROBO with a specific overlap size, e.g., ROBO-2. For all methods, we limit each experiment to 1000 iterations. One iteration is all selected agents performing one round of communication and optimization.

1) Chordal Cost Function: For the chordal distance PGO formulation, we compare ROBO with different levels of overlap against AMM-PGO$^\#$ [13] and RBCD++ [12], both state-of-the-art parallel, synchronous, and fully distributed approaches to PGO that also use the chordal distance objective. We run RBCD++ with greedy block selection and a fixed restart frequency of 30 iterations. We expect AMM-PGO$^\#$ to outperform RBCD++ per [13], but compare against RBCD++ as ROBO is also based on RBCD (without acceleration [12]).

We assess the performance of these methods using three initialization scenarios: *distributed chordal*, *block-wise spanning tree*, and *odometry*. These schemes are presented in increasing order of difficulty in terms of the resulting optimization, and decreasing order of complexity in terms of communication required. The selected techniques can all be performed without a central node. We use the distributed chordal initialization approach from [29] and the block-wise spanning tree method from [36]. In the third technique, each agent initializes from its local odometry measurements only.

The results of all three initialization scenarios for a relative suboptimality threshold Δ_{rel} of 0.1% are summarized in Table I. We consider the percentage of datasets on which each method has converged after 100, 500, and 1000 iterations. In all experiments shown, ROBO-ω for any ω and AMM-PGO$^\#$ reach convergence on more benchmark

TABLE I. SYNCHRONOUS ROBO AND BASELINE METHODS ON BENCHMARK DATASETS WITH VARIED INITIALIZATION. PERCENTAGE OF PROBLEMS SOLVED SHOWN AFTER 100, 500, 1000 ITERATIONS. COLORS SHOW BEST (GREEN) AND SECOND-BEST (YELLOW) RESULTS.

Method	**% Problems Solved at Iteration ($\Delta_{rel} \leq 0.1\%$)**								
	Dist. Chordal			Spanning Tree			Odometry		
	100	500	1000	100	500	1000	100	500	1000
RBCD++	45.5	45.5	59.1	13.6	27.3	31.8	0.00	9.1	22.7
AMM-PGO$^{\#}$	40.9	81.8	**90.9**	18.2	54.5	63.6	0.00	22.7	54.5
ROBO-1	45.5	72.7	81.8	9.1	40.9	45.5	9.1	18.2	27.3
ROBO-2	63.6	81.8	86.4	13.6	54.5	72.7	**13.6**	40.9	**63.6**
ROBO-3	**72.7**	**86.4**	**90.9**	**31.8**	**77.3**	**86.4**	**13.6**	**50.0**	59.1

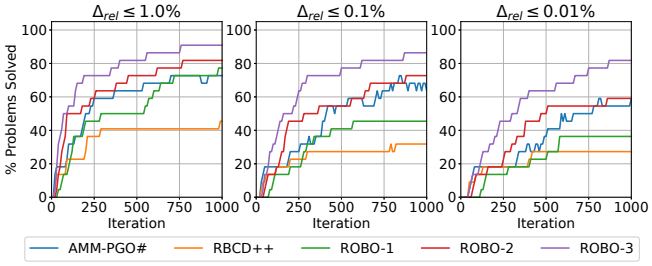


Fig. 2. Performance profile comparing AMM-PGO$^{\#}$, RBCD++, and ROBO with varying overlap levels when using block-wise spanning tree initialization. From left to right, the relative suboptimality thresholds used to determine convergence are 1%, 0.1%, and 0.01%.

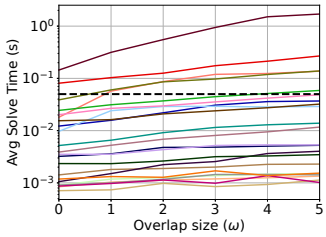
datasets within 1000 iterations than RBCD++. With distributed chordal initialization, ROBO-2 and ROBO-3 perform on par with AMM-PGO$^{\#}$. But for spanning tree and odometry initialization, which are easier to implement but result in a harder distributed optimization, ROBO-2 and ROBO-3 outperform both baselines. We also create a performance profile as in [13] showing the number of iterations required to reach a particular Δ_{rel}. Figure 2 shows the profile corresponding to block-wise spanning tree initialization for $\Delta_{rel} = 1\%,\ 0.1\%,\ 0.01\%$, from left to right. We note that due to the restart mechanism in both baseline methods, their corresponding performance lines can show oscillations. Figure 2 further highlights that for tighter Δ_{rel} thresholds, ROBO-2 and ROBO-3 also outperform both baselines. This trend is also present with odometry initialization, though we only include one profile here due to space limitations.

Table II enumerates the cost function value after 100 iterations for the three best-performing methods on individual datasets. Due to space constraints, we include only datasets with at least 5000 poses. Optimal costs are calculated using SE-Sync [33], and initial costs come from block-wise spanning tree initialization. We note that with this initialization, ROBO falls into local minima on `ais2klinik`, whereas our method converges on the same dataset with distributed chordal initialization. On all other large datasets, ROBO-2 and ROBO-3 perform better than the baselines.

2) Computation and Communication: The above results show that, in order to achieve faster convergence than state-of-the-art methods on benchmark datasets, ROBO requires $\omega \geq 2$. To assess the feasibility of using overlapping blocks of this size, we first consider the computation cost of locally solving larger subproblems. Figure 3a shows the per-robot,

TABLE II. COST AFTER 100 ITERATIONS OF ROBO AND BEST-PERFORMING BASELINE IN SYNCHRONOUS SETTING.

Dataset	**Opt.**	**Init.**	**Objective after 100 iterations**		
			AMM-PGO$^{\#}$	ROBO-2	ROBO-3
ais2klinik (2D)	188.5	2.6e8	**488**	1742	3.4e4
city (2D)	638.6	1.1e7	717.8	**638.6**	**638.6**
cubicle (3D)	717.1	2.5e6	852.9	**717.1**	**717.1**
grid (3D)	84319.01	4.9e5	84319.10	**84319.01**	**84319.01**
rim (3D)	5460.9	3.8e7	8702.3	5461.7	**5460.9**
torus (3D)	24227.05	6.3e4	24227.37	**24227.05**	**24227.05**



Dataset	**Poses**	**Edges**
grid (3D)	8000	22236
rim (3D)	10195	29743
cubicle (3D)	5750	16727
city (2D)	10000	20687
torus (3D)	5000	9048

(a) (b)

Fig. 3. (a) Local solve time as ω increases. Each curve shown is one dataset. For all but 4 of 22 datasets, local solve times are $\geq$ 20 Hz (dashed line) for $\omega \leq 3$. (b) Datasets with longest solve times (decreasing order).

single iteration computation time increase on all datasets with respect to ω as run on an Intel Xeon E-2276M 2.8 GHz processor with 32 GB of RAM. We observe the relative solve times between datasets, and the relative change in computation as ω increases, as these are primarily a function of graph structure and will be platform- and implementation-independent. The datasets with the five largest solve times are listed in Figure 3b. These datasets have larger graphs ($\geq$ 5000 poses) and are highly connected (i.e., many more edges than poses), which leads to greater relative increase in the domain size with larger ω than on a sparser graph. For the majority of the benchmarks, we observe mild or moderate increase in local solve time, demonstrating the practicality of ROBO when local computation is not extremely limited.

We next examine the robot-to-robot communication per iteration of ROBO relative to AMM-PGO$^{\#}$. We calculate data cost per iteration by counting the poses shared between each pair of robots, including prior poses. We consider both the average and maximum number of poses shared between all pairs of neighboring robots. Table III shows this average and maximum over all benchmark datasets, as well as the amount of data shared if each pose is represented by seven 32-bit floating point values. We also compare the number of iterations required to converge to $\Delta_{rel} = 0.1\%$ from distributed chordal initialization, noting "speedup" as a multiplier compared to the baseline method AMM-PGO$^{\#}$. We see that ROBO-2 and ROBO-3 offer an average speedup of $\mathbf{2.6}\times$ and $\mathbf{3.1}\times$ over AMM-PGO$^{\#}$ with an average of **only 25 Kb** additional per-iteration communication cost between robot pairs. In the worst case (corresponding to the `city` dataset), ROBO doubles the total network communication cost to roughly 250 Kb. Even with 100 iterations per second, this case would require 25 Mbps of bandwidth (versus the baseline of about 11 Mbps for the same iteration rate)—well within the limits of a 100 Mbps radio network like the

TABLE III. SPEEDUP AND ROBOT-TO-ROBOT COMMUNICATION LOAD OF ROBO-ω VS. AMM-PGO$^{\#}$ ACROSS ALL BENCHMARKS.

	AMM-PGO$^{\#}$	ROBO-2	ROBO-3
Avg. poses per iteration	50	130	159
Max. poses per iteration	481	1010	1153
Avg. data per iteration (Kb)	11	29	36
Max. data per iteration (Kb)	108	226	258
Avg. speedup over AMM-PGO$^{\#}$	-	**2.6**	**3.1**

TABLE IV. ROBO (GEODESIC COST) VS MESA-PARALLEL AND MESA

	Method	% Problems Solved at Iteration					
		$\Delta_{rel} \leq 0.1\%$			RMSE APE ≤ 0.1		
		100	500	1000	100	500	1000
Sync.	MESA-parallel	13.6	27.3	27.3	31.8	31.8	31.8
	ROBO-1	45.5	59.1	72.7	45.5	54.5	59.1
	ROBO-2	50.0	68.2	77.3	54.5	63.6	**72.7**
	ROBO-3	**59.1**	**72.7**	**81.8**	**59.1**	**68.2**	**72.7**
Edgewise	MESA	0.0	18.2	31.8	27.3	31.8	31.8
	ROBO-1	31.8	40.9	45.5	40.9	45.5	45.5
	ROBO-2	31.8	45.5	54.5	45.5	54.5	54.5
	ROBO-3	**40.9**	**59.1**	**63.6**	**50.0**	**59.1**	**59.1**

Silvus SL5200 [16]. Thus, considering both computation and communication, for high-bandwidth wireless networks often used in ground and air robotics applications, the additional data cost of ROBO for $\omega \leq 3$ is minor, even for large pose graphs. In practice, ω can be set to maximize available computation and communication, while considering that as previously noted, $\omega > 3$ yields diminishing returns.

3) Geodesic Cost Function: We compare ROBO with a geodesic distance cost against MESA [14], a state-of-the-art C-ADMM approach to DPGO that also uses the geodesic distance in its cost formulation [7]. For MESA, we use the hyperparameters the authors provide for benchmark dataset experiments. To match our parallel communication paradigm, we modify the original implementation of MESA[3] to allow all robots to optimize at every iteration in a synchronized fashion, which we denote as MESA-parallel. We use distributed chordal initialization in these comparisons. As tuning of the hyperparameters for MESA may affect the rate of decrease in the optimality gap, we also assess the RMSE APE. Once the RMSE APE is ≤ 0.1 m, we consider that dataset solved. Table IV (top) shows these summarized results for MESA-parallel and ROBO-ω with a geodesic distance cost function after 100, 500, and 1000 iterations. While ROBO with geodesic cost converges more slowly than with chordal cost, ROBO with any overlap converges more quickly than MESA-parallel in terms of both metrics.

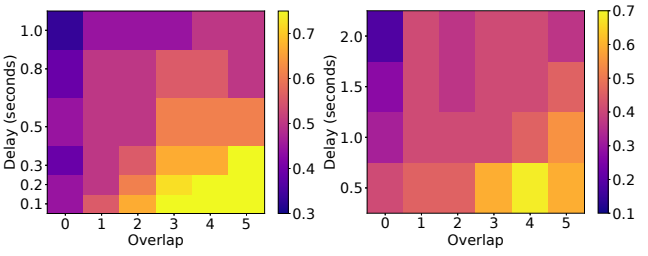
C. Edgewise ROBO

The edgewise-only paradigm restricts communication at each iteration to one pair of connected robots (see Section IV). We again use the geodesic cost function and distributed chordal initialization. We compare against the original implementation of MESA [14] with the same hyperparameters as in the previous experiments. The results summarized in Table IV (bottom) show that both in terms of relative suboptimality and RMSE APE, ROBO converges on more benchmarks than MESA within 1000 iterations. We see that as in the synchronized case, in this communicated-limited setting, sharing moderately more information between agents may significantly improve convergence rates.

[3]https://github.com/rpl-cmu/mesa

TABLE V. ASYNCHRONOUS ROBO COST (1S DELAY, $\lambda = 10$HZ)

Dataset	Opt.	Init.	ROBO-ω objective after 30 s				
			$\omega = 1$	2	3	4	5
ais2klinik (2D)	188.5	322	196.9	195.5	194.3	193.8	**192.6**
city (2D)	638.6	711	650.3	642.6	640.1	639.3	**638.8**
cubicle (3D)	717.1	835	720.2	719.0	718.0	749.5	**717.2**
grid (3D)	84319	87265	**84319**	84334	84324	84324	84436
rim (3D)	5460.9	8084	5890.2	5535.2	5469.3	5467.1	**5464.6**
torus (3D)	24227	24668	24228	24230	**24227**	**24227**	**24227**



(a) (b)

Fig. 4. Fraction of problems solved with $\Delta_{rel} \leq 0.1\%$ after 30 s with increasing delay and overlap ($\lambda = 10$ Hz). (a) Shorter delays, excluding 4 datasets with longest local solve times. (b) Longer delays, all datasets.

D. Asynchronous ROBO

Finally, we consider the asynchronous setting. In our experiments, we assess the sensitivity of ROBO to increasing delay d and overlap size ω. Each agent runs as its own process on a CPU. We use distributed chordal initialization before running all agents asynchronously for 30 seconds. We choose λ such that we can isolate the impact of the delays d on convergence. The solve times in Figure 3a show that for all but 4 datasets, $\lambda = 10$ Hz ensures that the expected value of the time between local optimizations is much larger than the local solve time. This means that for these 18 datasets, we can expect local optimization to run at approximately the rate λ. On this subset, we can test the impacts of shorter delays ($d = 0.1{-}1$ s) without local computation time adding to the delay. Figure 4a summarizes the fraction of these 18 benchmarks solved after 30 s for $\Delta_{rel} \leq 0.1\%$. Broadly, these results show that longer delays slow down convergence, and larger ω speeds up convergence for the same delay.

In Figure 4b, we repeat our experiments with all 22 datasets and $d \geq 0.5$ s. We choose larger jumps between delays to separate the effects of longer computation time on the largest datasets from increasing simulated delay. As in the synchronous setting, both heatmaps in Figure 4 show diminishing returns with increasing ω, with longer delays requiring greater increase in ω to achieve improved convergence rates. Additionally, for delays longer than 1 s, the convergence of ROBO is more sensitive to increasing overlap. Table V lists the objective value for datasets with ≥ 5000 poses after 30 s of asynchronous optimization with a communication delay of 1 s. We note that `grid`, which has

the longest local solve times for all ω, experiences greater delay than the simulated 1 s as ω increases, leading also to higher costs. For the remaining datasets, Table V highlights both the improvement from larger overlap, and the greater sensitivity of ROBO convergence when both delay and ω are large. These results show that ROBO with $\omega \leq 3$ is robust to moderate communication delays, and that overlap can be used to improve convergence even in asynchronous settings.

VI. Conclusion

This work presented ROBO, a novel, fully parallel distributed pose graph optimization method that utilizes available communication bandwidth for significantly improved convergence rates. We demonstrated the applicability of our method with varying levels of overlap in different initialization scenarios; with different rotation distance metrics in the objective; and in both synchronous and asynchronous settings on a wide range of benchmark datasets. We showed our method can achieve $3.1\times$ faster convergence over the state-of-the-art while requiring an average of only 36 Kb in per-iteration communication. Our future work will investigate a theoretical basis for both the convergence and limitations of our approach, explore conditions under which our method might have guaranteed convergence properties, and work toward a real-world asynchronous implementation. We will also consider the application of overlapping domain decomposition in other DPGO approaches, such as C-ADMM.

References

[1] K. Ebadi, L. Bernreiter, *et al.*, "Present and Future of SLAM in Extreme Environments: The DARPA SubT Challenge," *IEEE Trans. on Robotics*, vol. 40, pp. 936–959, 2024.

[2] Y. Chang, Y. Tian, J. P. How, and L. Carlone, "Kimera-Multi: a System for Distributed Multi-Robot Metric-Semantic Simultaneous Localization and Mapping," in *2021 IEEE Int. Conf. on Robotics and Automation*, 2021, pp. 11 210–11 218.

[3] P.-Y. Lajoie, B. Ramtoula, *et al.*, "DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams," *IEEE Robotics and Automation Letters*, vol. 5(2), pp. 1656–1663, Apr. 2020.

[4] Y. Chang, K. Ebadi, *et al.*, "LAMP 2.0: A Robust Multi-Robot SLAM System for Operation in Challenging Large-Scale Underground Environments," *IEEE Robotics and Automation Letters*, vol. 7(4), pp. 9175–9182, Oct. 2022.

[5] P. Schmuck, T. Ziegler, *et al.*, "COVINS: Visual-Inertial SLAM for Centralized Collaboration," in *2021 IEEE Int. Symposium on Mixed and Augmented Reality Adjunct*, Oct. 2021, pp. 171–176.

[6] F. Li, S. Yang, X. Yi, and X. Yang, "CORB-SLAM: A Collaborative Visual SLAM System for Multiple Robots," in *Collaborative Computing: Networking, Applications and Worksharing*. Springer International Publishing, 2018, pp. 480–490.

[7] F. Dellaert and GTSAM Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam

[8] S. Agarwal, K. Mierle, and The Ceres Solver Team, "Ceres Solver," Oct. 2023. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[9] S. Choudhary, L. Carlone, *et al.*, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *The Int. Journal of Robotics Research*, vol. 36(12), pp. 1286–1311, Oct. 2017.

[10] R. Murai, J. Ortiz, *et al.*, "A Robot Web for Distributed Many-Device Localization," *IEEE Trans. on Robotics*, vol. 40, pp. 121–138, 2024.

[11] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, "Asynchronous and Parallel Distributed Pose Graph Optimization," *IEEE Robotics and Automation Letters*, vol. 5(4), pp. 5819–5826, Oct. 2020.

[12] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, "Distributed Certifiably Correct Pose-Graph Optimization," *IEEE Trans. on Robotics*, vol. 37(6), pp. 2137–2156, Dec. 2021.

[13] T. Fan and T. D. Murphey, "Majorization Minimization Methods for Distributed Pose Graph Optimization," *IEEE Trans. on Robotics*, vol. 40, pp. 22–42, 2024.

[14] D. McGann, K. Lassak, and M. Kaess, "Asynchronous Distributed Smoothing and Mapping via On-Manifold Consensus ADMM," in *IEEE Int. Conf. on Robotics and Automation*, 2024, pp. 4577–4583.

[15] *Digi XBee 3 Zigbee 3.0 Datasheet*, Digi, 2023. [Online]. Available: https://www.digi.com/resources/library/data-sheets/ds_xbee-3-zigbee-3

[16] *StreamCaster LITE 5200 Data Sheet*, Silvus Technologies, 2024. [Online]. Available: https://silvustechnologies.com/wp-content/uploads/2024/10/StreamCaster-Lite-5200-SL5200-OEM-Module-Datasheet.pdf

[17] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2007.

[18] S. Shin, V. Zavala, and M. Anitescu, "Decentralized Schemes with Overlap for Solving Graph-Structured Optimization Problems," *IEEE Trans. on Control of Network Systems*, vol. 7(3), pp. 1225–1236, Sept. 2020.

[19] S. Shin, M. Anitescu, and V. Zavala, "Overlapping Schwarz Decomposition for Constrained Quadratic Programs," in *2020 59th IEEE Conf. on Decision and Control*, 2020, pp. 3004–3009.

[20] ——, "Exponential Decay of Sensitivity in Graph-Structured Nonlinear Programs," Dec. 2021, arXiv:2101.03067.

[21] L. A. A. Andersson and J. Nygards, "C-SAM: Multi-Robot SLAM using square root information smoothing," in *2008 IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 2798–2805.

[22] T. Bailey, M. Bryson, *et al.*, "Decentralised cooperative localisation for heterogeneous teams of mobile robots," in *2011 IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 2859–2865.

[23] Y. Zhang, M. Hsiao, *et al.*, "MR-iSAM2: Incremental Smoothing and Mapping with Multi-Root Bayes Tree for Multi-Robot SLAM," in *2021 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2021, pp. 8671–8678.

[24] X. Liu, J. Lei, *et al.*, "SlideSLAM: Sparse, Lightweight, Decentralized Metric-Semantic SLAM for Multi-Robot Navigation," July 2024, arXiv:2406.17249.

[25] P.-Y. Lajoie and G. Beltrame, "Swarm-SLAM : Sparse Decentralized Collaborative Simultaneous Localization and Mapping Framework for Multi-Robot Systems," *IEEE Robotics and Automation Letters*, vol. 9(1), pp. 475–482, Jan. 2024.

[26] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using Constrained Factor Graphs," in *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 3025–3030.

[27] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *2013 IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 5220–5227.

[28] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A Survey of Distributed Optimization Methods for Multi-Robot Systems," Mar. 2021, arXiv:2103.12840.

[29] T. Fan and T. Murphey, "Majorization Minimization Methods for Distributed Pose Graph Optimization with Convergence Guarantees," in *IEEE Int. Conf. on Int. Robots and Systems*, 2020, pp. 5058–5065.

[30] C. Li, G. Guo, P. Yi, and Y. Hong, "Distributed Pose-Graph Optimization With Multi-Level Partitioning for Multi-Robot SLAM," *IEEE Robotics and Automation Letters*, vol. 9(6), pp. 4926–4933, June 2024.

[31] T. H. Hamam and J. Romberg, "Streaming Solutions for Time-Varying Optimization Problems," *IEEE Trans. on Signal Processing*, vol. 70, pp. 3582–3597, 2022.

[32] S. Na, S. Shin, M. Anitescu, and V. Zavala, "On the Convergence of Overlapping Schwarz Decomposition for Nonlinear Optimal Control," *IEEE Trans. on Auto. Control*, vol. 67(11), pp. 5996–6011, Mar. 2022.

[33] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group," *The Int. Journal of Robotics Research*, vol. 38(2-3), pp. 95–125, 2019.

[34] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation Averaging," *Int. Journal of Computer Vision*, vol. 103, no. 3, pp. 267–305, July 2013.

[35] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM." https://github.com/MichaelGrupp/evo, 2017.

[36] Y. Tian and J. P. How, "Spectral Sparsification for Communication-Efficient Collaborative Rotation and Translation Estimation," *IEEE Trans. on Robotics*, vol. 40, pp. 257–276, 2024.