Majorization Minimization Methods for Distributed Pose Graph Optimization

Taosha Fan, Senior Member, IEEE, and Todd D. Murphey, Senior Member, IEEE

Abstract—We consider the problem of distributed pose graph optimization (PGO) that has important applications in multirobot simultaneous localization and mapping (SLAM). We propose the majorization minimization (MM) method for distributed PGO (MM–PGO) that applies to a broad class of robust loss kernels. The MM–PGO method is guaranteed to converge to first-order critical points under mild conditions. Furthermore, noting that the MM–PGO method is reminiscent of proximal methods, we leverage Nesterov’s method and adopt adaptive restarts to accelerate convergence. The resulting accelerated MM methods for distributed PGO—both with a master node in the network (AMM–PGO) and without (AMM–PGO*)—have faster convergence in contrast to the MM–PGO method without sacrificing theoretical guarantees. In particular, the AMM–PGO* method, which needs no master node and is fully decentralized, features a novel adaptive restart scheme and has a rate of convergence comparable to that of the AMM–PGO* method using a master node to aggregate information from all the nodes. The efficacy of this work is validated through extensive applications to 2-D and 3-D SLAM benchmark datasets and comprehensive comparisons against existing state-of-the-art methods, indicating that our MM methods converge faster and result in better solutions to distributed PGO.

Index Terms—Multi-robot systems, optimization methods, pose graph optimization, simultaneous localization and mapping.

I. INTRODUCTION

POSE graph optimization (PGO) is a nonlinear and nonconvex optimization problem estimating unknown poses from noisy relative pose measurements. PGO associates each pose with a vertex and each relative pose measurement with an edge such that the optimization problem is well represented through a graph. PGO has important applications in a number of areas, including but not limited to robotics [1], [2], [3], autonomous driving [4], and computational biology [5], [6]. Recent advances [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] suggest that PGO can be well solved using iterative optimization. Nevertheless, the aforementioned techniques [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] are difficult to distribute across a network due to communication and computational limitations and are only applicable to small- and medium-sized problems with at most tens of thousands poses. In addition, their centralized pipelines are equivalent to using a master node to aggregate information from the entire network and, thus, fail to meet privacy requirements one may wish to impose [17], [18].

In multirobot simultaneous localization and mapping (SLAM) [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], each robot estimates not only its own poses but those of the others as well to build an environment map. Even though such a problem can be solved by PGO, communication between robots is restricted, and multirobot SLAM has more unknown poses than single-robot SLAM. Thus, instead of using centralized PGO [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], it is more reasonable to formulate this large-sized estimation problem involving multiple robots as distributed PGO—each robot in multirobot SLAM is represented as a node and two nodes (robots) are said to be neighbors if there exists a noisy relative pose measurement between them (a more detailed description of distributed PGO can be found in Section IV). In most cases, it is assumed that internode communication only occurs between neighboring nodes and most of these iterative optimization methods are infeasible due to the expensive communication cost of solving linear systems and performing line search [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], which renders distributed PGO more challenging than centralized PGO.

In this article, we propose majorization minimization (MM) methods [29], [30] for distributed PGO. As the name would suggest, MM methods have two steps. First, in the majorization step, we construct a surrogate function that majorizes the objective function, i.e., the surrogate function is greater to the objective function except for the current iterate where both of them attain the same value. Then, in the minimization step, we minimize the surrogate function instead of the original objective function to improve the iterate. MM methods remain difficult, albeit straightforward, in practical use, e.g., the surrogate function, whose construction and minimization can not be more difficult than solving the optimization problem itself, is usually unknown, and MM methods might fail to converge and suffer from slow convergence. Thus, the implementation of MM methods on large-scale, complicated, and nonconvex optimization problems such as distributed PGO is nontrivial, and internode communication requirements impose extra restrictions making it more
so. All of these issues are addressed both theoretically and empirically in the rest of this article.

This article extends the preliminary results in [31] and [32], where we developed MM methods for centralized and distributed PGO that are guaranteed to converge to first-order critical points. In [31] and [32], we also introduced and elaborated on the use of Nesterov’s method [33], [34] and adaptive restart [35] for the first time to accelerate the convergence of PGO. Beyond the initial results in [31] and [32], this article presents completely redesigned MM methods for distributed PGO and provides more comprehensive theoretical and empirical results. In particular, our MM methods in this article are capable of handling a broad class of robust loss kernels, no longer require each iteration to attain a local optimal solution to the surrogate function for the convergence guarantees, and adopt a novel adaptive restart scheme for distributed PGO without a master node to make full use of Nesterov’s acceleration.

In summary, the contributions of this article are as follows.

1) We derive a class of surrogate functions that suit well with MM methods for distributed PGO. These surrogate functions apply to a broad class of robust loss kernels in robotics and computer vision.

2) We develop MM methods for distributed PGO that are guaranteed to converge to first-order critical points under mild conditions. Our MM methods for distributed PGO implement a novel update rule such that each iteration does not have to minimize the surrogate function to a local optimal solution.

3) We leverage Nesterov’s methods and adaptive restart to accelerate MM methods for distributed PGO and achieve significant improvement in convergence without any compromise of theoretical guarantees.

4) We present a decentralized adaptive restart scheme to make full use of Nesterov’s acceleration such that accelerated MM methods for distributed PGO without a master node are almost as fast as those requiring a master node.

The rest of this article is organized as follows. Section II reviews the state-of-the-art methods for distributed PGO. Section III introduces mathematical notation and preliminaries that are used in this article. Section IV formulates the problem of distributed PGO. Sections V and VI present surrogate functions for individual loss terms and the overall distributed PGO, respectively, which are fundamental to our MM methods. Sections VII–IX present unaccelerated and accelerated MM methods for distributed PGO that are guaranteed to converge to first-order critical points, which are the major contributions of this article. Section X implements our MM methods for distributed PGO on a number of simulated and real-world SLAM datasets and makes extensive comparisons against existing state-of-the-art methods [36], [37]. Finally, Section XI concludes this article and discusses future work.

II. RELATED WORK

In the last decade, multirobot SLAM has been becoming increasingly popular, which promotes the development of distributed PGO [36], [37], [38], [39]. Choudhary et al. [36] present a two-stage algorithm that implements either Jacobi overrelaxation or successive overrelaxation as distributed linear system solvers. Similar to centralized methods, Choudhary et al. [36] first evaluate the chordal initialization [40] and then improve the initial guess with a single Gauss–Newton step. However, one step of the Gauss–Newton method in most cases cannot lead to sufficient convergence for distributed PGO. In addition, no line search is performed in [36] due to the communication limitation, and thus, the behaviors of the single Gauss–Newton step is totally unpredictable and might result in bad solutions.

Tian et al. [37] present the distributed certifiably correct PGO using the Riemannian block coordinate descent method, which is later generalized to asynchronous and parallel distributed PGO [41]. Especially, their method makes use of Riemannian staircase optimization to solve the semidefinite relaxation of distributed PGO and is guaranteed to converge to global optimal solutions under moderate measurement noise. Following our previous works [31], [32], they implement Nesterov’s method for acceleration as well. Contrary to our MM methods, a major drawback of the work in [37] is that their method has to precompute red–black coloring assignment for block aggregation and keep part of the blocks in idle for estimate updates. In addition, although several strategies for block selection (e.g., greedy/importance sampling) and Nesterov’s acceleration (e.g., adaptive/fixed restarts) are adopted in [37] to improve the convergence, most of them are either inapplicable without a master node or at the sacrifice of computational efficiency and theoretical guarantees. In contrast, our MM methods are much faster (see Section X) but have no such restrictions for acceleration. More recently, Tian et al. [28] further apply Riemannian block coordinate descent method to distributed PGO with robust loss kernels. However, they solve robust distributed PGO by trivially updating the weights using graduated nonconvexity [42], and no formal proofs of convergence are provided. Again, this is in contrast to the work presented here that has provable convergence to first-order critical points for a broad class of robust loss kernels.

Tron and Vidal [38] present a consensus-based method for distributed PGO using the Riemannian gradient. The authors derive a condition for convergence guarantees related to the stepsize of the method and the degree of the pose graph. Nonetheless, their method estimates rotation and translation separately, fails to handle robust loss kernels, and needs extra computation to find the convergence-guaranteed stepsize.

Cristofalo et al. [39] present a novel distributed PGO method using Lyapunov theory and multiagent consensus. Their method is guaranteed to converge if the pose graph has certain topological structures. However, Cristofalo et al. [39] update rotations without exploiting the translational measurements and only apply to pairwise consistent PGO with nonrobust loss kernels.

In comparison to these aforementioned techniques, our MM methods have the mildest conditions (not requiring any specific pose graph structures, any extra computation for preprocessing, any master nodes for information aggregation, etc.) to converge to first-order critical points, apply to a broad class of robust loss kernels in robotics and computer vision, and manage to implement decentralized acceleration with convergence.
guarantees. Most importantly, as is shown in Section X, our MM methods outperform existing state-of-the-art methods in terms of both efficiency and accuracy on a variety of SLAM benchmark datasets.

III. NOTATION

Miscellaneous Sets: $\mathbb{R}$ denotes the sets of real numbers; $\mathbb{R}^+$ denotes the sets of nonnegative real numbers; $\mathbb{R}^{m \times n}$ and $\mathbb{R}^n$ denote the sets of $m \times n$ matrices and $n \times 1$ vectors, respectively. $SO(d)$ denotes the set of special orthogonal groups and $SE(d)$ denotes the set of special Euclidean groups. $| \cdot |$ denotes the cardinality of a set.

Matrices: For a matrix $X \in \mathbb{R}^{m \times n}$, $[X]_{ij}$ denotes the $(i, j)$th entry or $(i, j)$th block of $X$, and $[X]_i$ denotes the $i$th entry or $i$th block of $X$. For symmetric matrices $X, Y \in \mathbb{R}^{m \times n}$, $X \succeq Y$ (or $Y \preceq X$) and $X \succ Y$ (or $Y \prec X$) mean that $X - Y$ is positive (or negative) semidefinite and definite, respectively.

Inner Products and Norms: For a matrix $M \in \mathbb{R}^{m \times n}$, $\langle \cdot, \cdot \rangle_M : \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \to \mathbb{R}$ denotes the function

$$\langle X, Y \rangle_M \triangleq \text{trace}(XM^T)$$

(1)

where $X, Y \in \mathbb{R}^{m \times n}$. If $M$ is the identity matrix, $\langle \cdot, \cdot \rangle_M$ is also represented as $\langle \cdot, \cdot \rangle : \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \to \mathbb{R}$ such that

$$\langle X, Y \rangle \triangleq \text{trace}(XY^T).$$

(2)

For a positive semidefinite matrix $M \in \mathbb{R}^{n \times n}$, $\| \cdot \|_M : \mathbb{R}^{n \times n} \to \mathbb{R}^+$ denotes the function

$$\|X\|_M \triangleq \text{trace}(XM^T)$$

(3)

where $X \in \mathbb{R}^{n \times n}$. Also, $\| \cdot \|$ denotes the Frobenius norm of matrices and vectors, and $\| \cdot \|_2$ denotes the induced 2-norms of matrices and linear operators.

Riemannian Geometry: If $F(\cdot) : \mathbb{R}^{n \times n} \to \mathbb{R}$ is a function, $\mathcal{M} \subset \mathbb{R}^{n \times n}$ is a Riemannian manifold and $X \in \mathcal{M}$, then $\nabla F(X)$ and $\text{grad} F(X)$ denote the Euclidean and Riemannian gradients, respectively.

Graph Theory: PGO is represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ are the sets of vertices and edges, respectively [8]. In distributed PGO, each vertex is described as an ordered pair $(\alpha, i) \in \mathcal{V}$ where $\alpha$ is the node index and $i$ the local index of the vertex within node $\alpha$. For any nodes $\alpha$ and $\beta$ in distributed PGO, $\mathcal{E}^{\alpha \beta}$ denotes the set of edges between nodes $\alpha$ and $\beta$

$$\mathcal{E}^{\alpha \beta} \triangleq \{(i, j) | ((\alpha, i), (\beta, j)) \in \mathcal{E}\}$$

(4)

and $\mathcal{N}^\alpha$ denotes the set of nodes with edges from node $\alpha$

$$\mathcal{N}^\alpha \triangleq \{\beta | \mathcal{E}^{\alpha \beta} \neq \emptyset \text{ and } \alpha \neq \beta\}$$

(5)

and $\mathcal{N}_+^\alpha$ denotes the set of nodes with edges to node $\alpha$

$$\mathcal{N}_+^\alpha \triangleq \{\beta | \mathcal{E}^{\beta \alpha} \neq \emptyset \text{ and } \alpha \neq \beta\}$$

(6)

1A more complete summary of the notation is given in [43, Appendix A] and $\mathcal{N}^\alpha$ denotes the set of nodes with edges from or to node $\alpha$

$$\mathcal{N}^\alpha \triangleq \mathcal{N}_-^\alpha \cup \mathcal{N}_+^\alpha \triangleq \{\beta | \mathcal{E}^{\alpha \beta} \neq \emptyset \text{ or } \mathcal{E}^{\beta \alpha} \neq \emptyset \text{ and } \alpha \neq \beta\}.$$ (7)

Optimization: For optimization variables $X, X^\alpha, R^\alpha, t^\alpha$, etc., the notation $X^{(k)}, X^{(k)}^\alpha, R^{(k)}_\alpha, t^{(k)}_\alpha$, etc., denotes the $k$th iterate of corresponding optimization variables.

IV. PROBLEM FORMULATION

A. Distributed PGO

In distributed PGO [36], [37], [38], we are given $|\mathcal{A}|$ nodes $\mathcal{A} \triangleq \{1, 2, \ldots, |\mathcal{A}|\}$ and each node $\alpha \in \mathcal{A}$ has $n_\alpha$ poses $g^\alpha_1, g^\alpha_2, \ldots, g^\alpha_{n_\alpha} \in SE(d)$. Let $g^\alpha_i \triangleq (t^\alpha_i, R^\alpha_i)$ where $t^\alpha_i \in \mathbb{R}^d$ is the translation and $R^\alpha_i \in SO(d)$ the rotation. We consider the problem of estimating unknown poses $g^\alpha_1, g^\alpha_2, \ldots, g^\alpha_{n_\alpha} \in SE(d)$ for all the nodes $\alpha \in \mathcal{A}$ given intranode noisy measurements $\tilde{g}^{\alpha \beta}_{ij} = (\tilde{t}^{\alpha \beta}_{ij}, \tilde{R}^{\alpha \beta}_{ij}) \in SE(d)$ of the relative pose

$$g^{\alpha \beta}_{ij} = (g^\alpha_i)^{-1} g^\beta_j \in SE(d)$$

(8)

within a single node $\alpha$, and internode noisy measurements $\tilde{g}^{\alpha \beta}_{ij} = (\tilde{t}^{\alpha \beta}_{ij}, \tilde{R}^{\alpha \beta}_{ij}) \in SE(d)$ of the relative pose

$$g^{\alpha \beta}_{ij} = (g^\alpha_i)^{-1} g^\beta_j \in SE(d)$$

(9)

between different nodes $\alpha \neq \beta$. In (8) and (9), note that $\tilde{t}^{\alpha \beta}_{ij}$ and $\tilde{R}^{\alpha \beta}_{ij}$ are translational measurements, and $\tilde{R}^{\alpha \beta}_{ij}$ and $\tilde{R}^{\alpha \beta}_{ij}$ are rotational measurements.

Following (4)–(7), we represent distributed PGO as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that unknown pose $g^\alpha_i \in SE(d)$ and noisy measurement $\tilde{g}^{\alpha \beta}_{ij} \in SE(d)$ have one-to-one correspondence to vertex $(\alpha, i) \in \mathcal{V}$ and directed edge $((\alpha, i), (\beta, j)) \in \mathcal{E}$, respectively. We refer nodes $\alpha$ and $\beta \in \mathcal{A}$ as neighbors as long as either $\mathcal{E}^{\alpha \beta} \neq \emptyset$ or $\mathcal{E}^{\beta \alpha} \neq \emptyset$. Then, $\mathcal{N}^\alpha$ and $\mathcal{N}_+^\alpha$ are the sets of neighbors with a directed edge from and to node $\alpha$, respectively, and $\mathcal{N}^\alpha$ is the set of neighbors with a directed edge connected to node $\alpha$.

In the rest of this article, we make the following assumption that each node $\alpha$ can communicate with its neighbors and the network topology is unchanged during optimization. These assumptions are common in distributed PGO [36], [37], [38], [39].

Assumption 1: Each node $\alpha$ can communicate with its neighbors $\beta \in \mathcal{N}^\alpha$ and the network topology is fixed.

B. Loss Kernels

In practice, it is inevitable that there exist internode measurements that are outliers resulting from false loop closures. These outliers adversely affect the overall performance of distributed PGO. To address this issue, it is popular to use nontrivial loss kernels—e.g., Huber and Welsch losses—to enhance the robustness of distributed PGO [44], [45], [46].
In this article, we make the following assumption that applies to a broad class of loss kernels \( \rho(\cdot) : \mathbb{R}^+ \to \mathbb{R} \) in robotics and computer vision.

**Assumption 2:** The loss kernel \( \rho(\cdot) : \mathbb{R}^+ \to \mathbb{R} \) satisfies the following properties.

1. \( \rho(s) \geq 0 \) for any \( s \in \mathbb{R}^+ \) and the inequality “\( = \)” holds if and only if \( s = 0 \).
2. \( \rho(\cdot) : \mathbb{R}^+ \to \mathbb{R} \) is continuously differentiable.
3. \( \rho(\cdot) : \mathbb{R}^+ \to \mathbb{R} \) is a concave function.
4. \( 0 \leq \nabla \rho(s) \leq 1 \) for any \( s \in \mathbb{R}^+ \) and \( \nabla \rho(0) = 1 \).
5. \( \varphi(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R} \) with \( \varphi(X) \triangleq \rho(\|X\|^2) \) has Lipschitz continuous gradient, i.e., there exists \( \mu > 0 \) such that \( \| \nabla \varphi(X) - \nabla \varphi(X') \| \leq \mu \cdot \| X - X' \| \) for any \( X, X' \in \mathbb{R}^{m \times n} \).

In the following, we present some examples of loss kernels (see Fig. 1) satisfying Assumption 2.

**Example 1 (Trivial Loss):**

\[
\rho(s) = s. \tag{10}
\]

**Example 2 (Huber Loss):**

\[
\rho(s) = \begin{cases} 
  s, & |s| \leq a \\
  2\sqrt{\alpha|s|} - a, & |s| > a 
\end{cases} \tag{11}
\]

where \( \alpha > 0 \).

**Example 3 (Welsch Loss):**

\[
\rho(s) = a - a \exp \left( -\frac{s}{\alpha} \right) \tag{12}
\]

where \( \alpha > 0 \).

**C. Objective Function**

Recall that each node \( \alpha \in A \) has \( n_\alpha \) unknown poses \( g^{(1)}_\alpha, g^{(2)}_\alpha, \ldots, g^{(n_\alpha)}_\alpha \in \text{SE}(d) \). For notational simplicity, we define \( X^\alpha \) and \( X \) as

\[
X^\alpha \triangleq \mathbb{R}^{d \times n_\alpha} \times \text{SO}(d)^{n_\alpha}
\]

and

\[
X \triangleq X^1 \times \cdots \times X^{|A|} \subset \mathbb{R}^{d \times (d+1)^n}
\]

respectively, where \( n \triangleq \sum_{\alpha \in A} n_\alpha \). Furthermore, we represent \( g^{(i)}_\alpha \in \text{SE}(d) \), i.e., the \( i \)-th pose of node \( \alpha \in A \), as a \( d \times (d+1) \)

matrix

\[
X^\alpha \triangleq [t^\alpha_i R^\alpha_i] \in \text{SE}(d) \subset \mathbb{R}^{d \times (d+1)}
\]

represent \( \{g^{(1)}_\alpha, g^{(2)}_\alpha, \ldots, g^{(n_\alpha)}_\alpha \} \in \text{SE}(d)^{n_\alpha} \), i.e., all the poses of node \( \alpha \in A \), as an element of \( \mathcal{X}^\alpha \) as well as a \( d \times (d+1)n_\alpha \)

matrix

\[
X^\alpha \triangleq [t^\alpha_i R^\alpha_i] \in \mathcal{X}^\alpha \subset \mathbb{R}^{d \times (d+1)n_\alpha}
\]

where

\[
t^\alpha_i \triangleq [t^\alpha_i \ldots t^{n_\alpha}_i] \in \mathbb{R}^{d \times n_\alpha}
\]

and

\[
R^\alpha_i \triangleq [R^\alpha_{i1} \ldots R^\alpha_{in_\alpha}] \in \text{SO}(d)^{n_\alpha} \subset \mathbb{R}^{d \times dn_\alpha}
\]

and represent \( \{g^{(1)}_\alpha, g^{(2)}_\alpha, \ldots, g^{(n_\alpha)}_\alpha \} \in \text{SE}(d)^{n_\alpha} \), i.e., all the poses of distributed PGO, as an element of \( \mathcal{X} \) as well as a \( d \times (d+1)n \)

matrix

\[
X \triangleq \begin{bmatrix} X^1 & \cdots & X^{|A|} \end{bmatrix} \in \mathcal{X} \subset \mathbb{R}^{d \times (d+1)n_\alpha}
\]

**Remark 1:** \( \mathcal{X}^\alpha \) and \( \mathcal{X} \) are by definition homeomorphic to \( \text{SE}(d)^{n_\alpha} \) and \( \text{SE}(d)^n \), respectively. Thus, \( X^\alpha \in \mathcal{X}^\alpha \) and \( X \in \mathcal{X} \) are sufficient to represent elements of \( \text{SE}(d)^{n_\alpha} \) and \( \text{SE}(d)^n \).

Following [8], [31], and [32], distributed PGO can be formulated as an optimization problem on \( X = \begin{bmatrix} X^1 & \cdots & X^{|A|} \end{bmatrix} \in \mathcal{X} \):

**Problem 1 (Distributed PGO):**

\[
\min_{X \in \mathcal{X}} F(X). \tag{16}
\]

The objective function \( F(X) \) in (16) is defined as

\[
F(X) \triangleq \sum_{\alpha \in A} \sum_{(i,j) \in \mathcal{E}^\alpha} \frac{1}{2} \left( \kappa^{\alpha}_{ij} \| R^\alpha_i R^\alpha_j^\alpha - R_j^\alpha \|^2 
+ \tau^{\alpha}_{ij} \| R^\alpha_i R^\alpha_j^\alpha + t^\alpha_i - t^\alpha_j \|^2 \right) 
+ \sum_{\alpha \in A} \sum_{(i,j) \in \mathcal{E}^\alpha} \frac{1}{2} \left( \kappa^{\alpha \beta}_{ij} \| R^\alpha_i R^\alpha_j^\alpha - R_j^\beta \|^2 
+ \tau^{\alpha \beta}_{ij} \| R^\alpha_i R^\alpha_j^\alpha + t^\alpha_i - t^\beta_j \|^2 \right) \tag{17}
\]

where \( \kappa^{\alpha}_{ij}, \tau^{\alpha}_{ij}, \kappa^{\alpha \beta}_{ij}, \) and \( \tau^{\alpha \beta}_{ij} \) are the weights and \( \rho(\cdot) : \mathbb{R}^+ \to \mathbb{R} \) is the loss kernel.

For notational simplicity, \( F(X) \) in (17) can be also rewritten as

\[
F(X) = \sum_{\alpha \in A} \sum_{(i,j) \in \mathcal{E}^\alpha} F^{\alpha}_{ij}(X) 
+ \sum_{\alpha \neq \beta \in A} \sum_{(i,j) \in \mathcal{E}^\alpha} F^{\alpha \beta}_{ij}(X) \tag{18}
\]

where

\[
F^{\alpha}_{ij}(X) \triangleq \frac{1}{2} \kappa^{\alpha}_{ij} \| R^\alpha_i R^\alpha_j^\alpha - R_j^\alpha \|^2 
+ \frac{1}{2} \tau^{\alpha}_{ij} \| R^\alpha_i R^\alpha_j^\alpha + t^\alpha_i - t^\alpha_j \|^2 \tag{19a}
\]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
\[ F_{ij}^{\alpha \beta}(X) \triangleq \frac{1}{2} \rho \left( \kappa_{ij}^{\alpha \beta} R_{ij}^{\alpha} - R_{ij}^{\beta} \right)^2 + \frac{1}{2} \tau_{ij}^{\alpha \beta} \left( R_{ij}^{\alpha} t_{ij}^\alpha + t_{ij}^\beta - t_{ij}^\beta \right)^2. \]  

(19b)

Note that \( F_{ij}^{\alpha \alpha}(X) \) and \( F_{ij}^{\alpha \beta} \) corresponds to intra- and internode measurements, respectively.

In the next sections, we will present MM methods for distributed PGO, which is the major contribution of this article.

V. MAJORIZATION OF LOSS KERNELS

In this section, we present surrogate functions majorizing the loss kernels \( \rho(\cdot) \). The resulting surrogate functions lead to an intermediate upper bound of distributed PGO while attaining the same value as the original objective function at each iterate.

It is straightforward to show that there exists sparse and positive semidefinite matrices \( M_{ij}^{\alpha \beta} \in \mathbb{R}^{(d+1)n \times (d+1)n} \) for either \( \alpha = \beta \) or \( \alpha \neq \beta \) such that

\[
\frac{1}{2} \| X \|_{M_{ij}^{\alpha \beta}}^2 = \frac{1}{2} \| R_{ij}^{\alpha} - R_{ij}^{\beta} \|^2 + \frac{1}{2} \tau_{ij}^{\alpha \beta} \| R_{ij}^{\alpha} t_{ij}^\alpha + t_{ij}^\beta - t_{ij}^\beta \|^2. 
\]

(20)

Then, in terms of intranode measurements with \( \alpha = \beta \) and internode measurements with \( \alpha \neq \beta \), \( F_{ij}^{\alpha \alpha}(X) \) and \( F_{ij}^{\alpha \beta} \) take the form of

\[
F_{ij}^{\alpha \alpha}(X) = \frac{1}{2} \| X \|_{F_{ij}^{\alpha \alpha}}^2 
\]

(21a)

\[
F_{ij}^{\alpha \beta}(X) = \frac{1}{2} \rho \left( \| X \|_{F_{ij}^{\alpha \beta}}^2 \right). 
\]

(21b)

From (19a) and (19b), we obtain an upper bound of \( F_{ij}^{\alpha \alpha}(X) \) and \( F_{ij}^{\alpha \beta}(X) \) as the following proposition states.

Proposition 1: Let \( X^{(k)} = [X^{1(k)} \cdots X^{n(k)}] \in \mathcal{X} \) with \( X^{(k)} \in \mathcal{X}^{(k)} \) be an iterate of (16). If \( \rho(\cdot) : \mathbb{R}^+ \to \mathbb{R} \) is a loss kernel that satisfies Assumption 2, then we obtain

\[
\frac{1}{2} \omega_{ij}^{\alpha \beta}(k) \| X - X^{(k)} \|_{M_{ij}^{\alpha \beta}}^2 + \langle \nabla F_{ij}^{\alpha \beta}(X^{(k)}), X - X^{(k)} \rangle 
\]

\[
+ F_{ij}^{\alpha \beta}(X^{(k)}) \geq F_{ij}^{\alpha \beta}(X) 
\]

(22)

for any \( X \) and \( X^{(k)} \in \mathbb{R}^{d \times (d+1)n} \), in which \( \omega_{ij}^{\alpha \beta}(k) \in \mathbb{R} \) is defined as

\[
\omega_{ij}^{\alpha \beta}(k) \triangleq \begin{cases} 
1, & \alpha = \beta \\
\nabla \rho \left( \| X^{(k)} \|_{M_{ij}^{\alpha \beta}}^2 \right), & \alpha \neq \beta. 
\end{cases} 
\]

(23)

In (22), the inequality “\( \geq \)” holds as long as \( X = X^{(k)} \).

Proof: See [43, Appendix B].

A. Majorization of \( F_{ij}^{\alpha \beta}(X) \)

For any matrices \( B, C, \) and \( P \in \mathbb{R}^{m \times n} \), it can be shown that

\[
\frac{1}{2} \| B - C \|_{M_{ij}^{\alpha \beta}}^2 \leq \| B - P \|_{M_{ij}^{\alpha \beta}}^2 + \| C - P \|_{M_{ij}^{\alpha \beta}}^2 
\]

(26)

as long as \( M_{ij}^{\alpha \beta} \in \mathbb{R}^{n \times n} \) is positive semidefinite, where “\( \geq \)” holds if

\[
P = \frac{1}{2} B + \frac{1}{2} C. 
\]

If we let \( P = 0 \), (26) becomes

\[
\frac{1}{2} \| B - C \|_{M_{ij}^{\alpha \beta}}^2 \leq \| B \|_{M_{ij}^{\alpha \beta}}^2 + \| C \|_{M_{ij}^{\alpha \beta}}^2 
\]

(27)

which holds for any \( B \) and \( C \in \mathbb{R}^{m \times n} \). Applying (27) on the right-hand side of (20), we obtain

\[
\frac{1}{2} \| X \|_{M_{ij}^{\alpha \beta}}^2 \leq \kappa_{ij}^{\alpha \beta} \| R_{ij}^{\alpha} \|_{M_{ij}^{\alpha \beta}}^2 + \kappa_{ij}^{\alpha \beta} \| R_{ij}^{\beta} \|_{M_{ij}^{\alpha \beta}}^2 + \tau_{ij}^{\alpha \beta} \| R_{ij}^{\alpha} t_{ij}^\alpha + t_{ij}^\beta \|_{M_{ij}^{\alpha \beta}}^2 + \tau_{ij}^{\alpha \beta} \| t_{ij}^\beta \|_{M_{ij}^{\alpha \beta}}^2 
\]

\[
= \kappa_{ij}^{\alpha \beta} \| R_{ij}^{\alpha} \|_{M_{ij}^{\alpha \beta}}^2 + \kappa_{ij}^{\alpha \beta} \| R_{ij}^{\beta} \|_{M_{ij}^{\alpha \beta}}^2 + \tau_{ij}^{\alpha \beta} \| R_{ij}^{\alpha} t_{ij}^\alpha + t_{ij}^\beta \|_{M_{ij}^{\alpha \beta}}^2 + \tau_{ij}^{\alpha \beta} \| t_{ij}^\beta \|_{M_{ij}^{\alpha \beta}}^2 
\]

(28)

where the last equality is due to \( (R_{ij}^{\alpha})^\top R_{ij}^{\alpha \beta} = \bar{R}_{ij}^{\alpha \beta} \bar{R}_{ij}^{\alpha \beta} = \mathbf{I} \).

Furthermore, there exists a positive semidefinite matrix \( \Omega_{ij}^{\alpha \beta} \in \mathbb{R}^{(d+1)n \times (d+1)n} \) such that the right-hand side of (28) can be
rewritten as
\[
\frac{1}{2} \| X \|_{\Omega_{ij}}^2 = \kappa_{ij}^{\alpha\beta} \| R^\alpha_{ij} \|_F^2 + \kappa_{ij}^{\alpha\beta} \| R^\beta_{ij} \|_F^2 \\
+ \tau_{ij}^{\alpha\beta} \| R^\alpha_{ij} G^\beta_{ij} + t^\alpha \|_F^2 + \tau_{ij}^{\alpha\beta} \| G^\beta_{ij} \|_F^2
\]
where \( \Omega_{ij}^{\alpha\beta} \) is a block diagonal matrix decoupling unknown poses of different nodes. Replacing the right-hand side of (28) with (29) results in
\[
\frac{1}{2} \| X \|_{M_{ij}}^2 \leq \frac{1}{2} \| X \|_{\Omega_{ij}^{\alpha\beta}}^2
\]
for any \( X \in \mathbb{R}^{d \times (d+1)n} \), which suggests
\[
\Omega_{ij}^{\alpha\beta} \succeq M_{ij}^{\alpha\beta}.
\]

With \( \Omega_{ij}^{\alpha\beta} \in \mathbb{R}^{(d+1)n \times (d+1)n} \) in (29) and (30), we define
\[
E_{ij}^{\alpha\beta}(\cdot | X(k)) \colon \mathbb{R}^{d \times (d+1)n} \to \mathbb{R}.
\]
\[
E_{ij}^{\alpha\beta}(X|X(k)) \triangleq \frac{1}{2} \omega_{ij}^{\alpha\beta}( \| X - X(k) \|_{\Omega_{ij}^{\alpha\beta}}^2 \\
+ \langle \nabla F_{ij}^{\alpha\beta}(X(k)), X - X(k) \rangle + F_{ij}^{\alpha\beta}(X(k))
\]
where \( \omega_{ij}^{\alpha\beta}(k) \) is given in (23). From the equation above, it can be concluded that \( E_{ij}^{\alpha\beta}(X|X(k)) \) majorizes \( F_{ij}^{\alpha\beta}(X) \) as the following proposition states, which is important for the construction of surrogate functions for distributed PGO.

**Proposition 2:** Given any nodes \( \alpha, \beta \in A \) with either \( \alpha = \beta \) or \( \alpha \neq \beta \), if \( \rho(\cdot) \colon \mathbb{R}^+ \to \mathbb{R} \) is a loss kernel that satisfies Assumption 2, then we obtain
\[
E_{ij}^{\alpha\beta}(X|X(k)) \geq F_{ij}^{\alpha\beta}(X).
\]
for any \( X \in \mathbb{R}^{d \times (d+1)n} \). In the previous equation, the equality \( \text{“} = \text{”} \) holds if \( X = X(k) \).

**Proof:** See [43, Appendix C]. \( \square \)

### B. Majorization of \( F(X) \)

From Proposition 2, it is straightforward to construct surrogate functions majorizing \( F(X) \) in (17) as the following proposition states.

**Proposition 3:** Let \( X(k) = [X^{1(k)}, \ldots, X^{d(k)}] \in \mathcal{X} \) with \( X^{\alpha}(k) \in \mathcal{X}^{\alpha} \) be an iterate of \( X \in \mathcal{X} \) in (16). Suppose \( G(\cdot | X(k)) \colon \mathbb{R}^{d \times (d+1)n} \to \mathbb{R} \) is a function
\[
G(X|X(k)) \triangleq \sum_{\alpha \in A} \sum_{(i,j) \in \mathcal{E}^{\alpha\beta}} E_{ij}^{\alpha\beta}(X|X(k)) \\
+ \sum_{\alpha,\beta \in A, \alpha \neq \beta} \sum_{(i,j) \in \mathcal{E}^{\alpha\beta}} E_{ij}^{\alpha\beta}(X|X(k)) \\
+ \frac{\xi}{2} \| X - X(k) \|_F^2
\]
where \( \xi \in \mathbb{R} \) and \( \xi \geq 0 \). Then, we have the following results.
1) For any \( X \in \mathbb{R}^{d \times (d+1)n} \) and \( X(k) \in \mathcal{X} \)
\[
G(X|X(k)) \geq F(X)
\]
where the equality \( \text{“} = \text{”} \) holds if \( X = X(k) \).
2) \( G(X|X(k)) \) is equivalent to
\[
G(X|X(k)) = \sum_{\alpha \in A} G^\alpha(X^{\alpha}|X(k)) + F(X(k))
\]
where \( G^\alpha(X^{\alpha}|X(k)) \) is a function of \( X^{\alpha} \in \mathcal{X}^{\alpha} \) within a single node \( \alpha \).
3) For any node \( \alpha \in A \) and \( i \in \{1, \ldots, n_\alpha\} \), there exists positive-semidefinite matrices \( \Pi_i^{\alpha}(k) \in \mathbb{R}^{(d+1)n_\alpha \times (d+1)n_\alpha} \) such that
\[
G^\alpha(X^{\alpha}|X(k)) \triangleq \frac{1}{2} \| X^{\alpha} - X^{\alpha}(k) \|_F^2 \\
+ \langle \nabla X^{\alpha} F(X(k)), X^{\alpha} - X^{\alpha}(k) \rangle
\]
where \( \nabla X^{\alpha} F(X(k)) \) is the Euclidean gradient of \( F(X) \) with respect to \( X^{\alpha} \in \mathcal{X}^{\alpha} \) at \( X(k) \in \mathcal{X} \).

**Proof:** See [43, Appendix D].

Following Proposition 3, we might further majorize \( F(X) \) as well as \( G(X|X(k)) \) by applying (32) to (33) and replacing \( F^{\alpha\beta}_{ij}(X|X(k)) \) with \( E_{ij}^{\alpha\beta}(X|X(k)) \), which results in the following proposition.

**Proposition 4:** Let \( X(k) = [X^{1(k)}, \ldots, X^{d(k)}] \in \mathcal{X} \) with \( X^{\alpha}(k) \in \mathcal{X}^{\alpha} \) be an iterate of \( X \in \mathcal{X} \) in (16), and \( X^{\alpha}(k) = [\rho^\alpha(k), \rho^{\alpha\beta}(k)] \in \mathcal{E}(d) \) the corresponding iterate of \( X^{\alpha} \in \mathcal{SE}(d) \). Suppose \( H(\cdot | X) \colon \mathbb{R}^{d \times (d+1)n} \to \mathbb{R} \) is a function
\[
H(X|X(k)) = \sum_{\alpha \in A} \sum_{(i,j) \in \mathcal{E}^{\alpha\beta}} E_{ij}^{\alpha\beta}(X|X(k)) \\
+ \sum_{\alpha,\beta \in A, \alpha \neq \beta} \sum_{(i,j) \in \mathcal{E}^{\alpha\beta}} E_{ij}^{\alpha\beta}(X|X(k)) \\
+ \frac{\zeta}{2} \| X - X(k) \|_F^2
\]
where \( \zeta \in \mathbb{R} \) and \( \zeta \geq 0 \). Note that \( \xi \in \mathbb{R} \) is given in (33). Then, we have the following results.
1) For any \( X \in \mathbb{R}^{d \times (d+1)} \) and \( X(k) \in \mathcal{X} \)
\[
H(X|X(k)) \geq G(X|X(k)) \geq F(X)
\]
where the equality \( \text{“} = \text{”} \) holds if \( X = X(k) \).
2) \( H(X|X(k)) \) is equivalent to
\[
H(X|X(k)) = \sum_{\alpha \in A} H^\alpha(X^{\alpha}|X(k)) + F(X(k)) \\
= \sum_{\alpha \in A} \sum_{i=1}^{n_\alpha} \Pi_i^{\alpha}(X_i^{\alpha}, X^{\alpha}|X(k)) + F(X(k))
\]
where \( H^\alpha(X^{\alpha}|X(k)) \) is a function of \( X^{\alpha} \in \mathcal{X}^{\alpha} \) within a single node \( \alpha \), and \( H^\alpha(X_i^{\alpha}|X(k)) \) is a function of a single pose \( X_i^{\alpha} \in \mathcal{SE}(d) \subset \mathbb{R}^{d \times (d+1)} \).
3) For any node \( \alpha \in A \) and \( i \in \{1, \ldots, n_\alpha\} \), there exists positive-semidefinite matrices \( \Pi_i^{\alpha}(k) \in \mathbb{R}^{(d+1)n_\alpha \times (d+1)n_\alpha} \) and \( \Pi_i^{\alpha}(k) \in \mathbb{R}^{(d+1)n_\alpha \times (d+1)n_\alpha} \) such that
that
\[ H^\alpha(X^\alpha|X^{(k)}) = \frac{1}{2} \|X^\alpha - X^{\alpha(k)}\|_F^2 + \langle \nabla X^\alpha F(X^{(k)}), X^\alpha - X^{\alpha(k)} \rangle \]
\[ H^i_1(X^{\alpha_1}|X^{(k)}) = \frac{1}{2} \|X^{i_1} - X^{i_1(k)}\|_F^2 + \langle \nabla X^{i_1} F(X^{(k)}), X^{i_1} - X^{i_1(k)} \rangle \]
where \( \nabla X^\alpha F(X^{(k)}) \) and \( \nabla X^{i_1} F(X^{(k)}) \) are the Euclidean gradients of \( F(X) \) with respect to \( X^\alpha \in X^\alpha \) and \( X^{i_1} \in \) \( \text{SE}(d) \) at \( X^{(k)} \) in \( \mathcal{X} \), respectively.

**Proof:** The proof is similar to that of Proposition 3.

**Remark 1:** As a result of (39), \( H^\alpha(X^\alpha|X^{(k)}) \) can be rewritten as the sum of \( H^i_1(X^{i_1}|X^{(k)}) \), i.e.,
\[ H^\alpha(X^\alpha|X^{(k)}) = \sum_{i=1}^{n_\alpha} H^i_1(X^{i_1}|X^{(k)}). \]

Note that \( H^i_1(X^{i_1}|X^{(k)}) \) in (39) and (42) relies on a single pose \( X^{i_1} \in \text{SE}(d) \subseteq \mathbb{R}^{d \times (d+1)} \). This will be later exploited in Sections VII-IX to improve the computational efficiency of distributed PGO.

**Remark 4:** Propositions 3 and 4 indicate that \( G(X|X^{(k)}) \) and \( H(X|X^{(k)}) \) not only majorize \( F(X) \) but also decouple poses from different nodes through \( G^\alpha(X^\alpha|X^{(k)}) \) and \( H^i_1(X^{i_1}|X^{(k)}) \), making it possible to implement MM methods for distributed PGO.

**Remark 5:** \( \zeta \) and \( \xi \) in (33) and (37) are important for the convergence analysis of MM methods for distributed PGO in Sections VII-IX. It is recommended to be set \( \zeta > 0 \) but close to zero such that \( G(X|X^{(k)}) \) and \( H(X|X^{(k)}) \) are tighter upper bounds of \( F(X) \) and yield faster convergence.

Recall that \( G^\alpha(X^\alpha|X^{(k)}) \) and \( H^i_1(X^{i_1}|X^{(k)}) \) in (36) and (40) rely on \( \Gamma^\alpha \), \( \Pi^{\alpha(k)} \), \( \nabla X^\alpha F(X^{(k)}) \), which—according to (18) and (19)—are only related to \( F^{\alpha(i)}_{ij}(X) \), \( F^{\alpha(i)}_{ij}(X) \), and \( F^{2\alpha(i)}_{ij}(X) \). Also, (19) indicates that \( F^{\alpha(i)}_{ij}(X) \) depends on \( X^{i_1} \) and \( X^{\beta_1} \), while \( F^{2\alpha(i)}_{ij}(X) \) and \( F^{3\alpha(i)}_{ij}(X) \) on \( X^{i_1} \) and \( X^{\beta_1} \). Therefore, \( \Pi^{\alpha(k)} \), \( \Gamma^\alpha \), \( \nabla X^\alpha F(X^{(k)}) \) can be evaluated as long as node \( \alpha \) has access to \( X^{\beta_1} \) from its neighbor \( \beta \). Furthermore, \( G^\alpha(X^\alpha|X^{(k)}) \) and \( H^i_1(X^{i_1}|X^{(k)}) \) can be constructed in a distributed setting with one internode communication round between neighboring nodes \( \alpha \) and \( \beta \).

In the next sections, we will present MM methods for distributed PGO using \( G(X|X^{(k)}) \) and \( H(X|X^{(k)}) \) that are guaranteed to converge to first-order critical points.

**VII. MM METHOD FOR DISTRIBUTED PGO**

In distributed optimization, MM methods are one of the most popular first-order optimization methods [29], [30]. As mentioned before, MM methods solve an optimization problem by iteratively minimizing an upper bound of the objective function such that the objective value is nonincreasing. Recall that \( G(X|X^{(k)}) \) and \( H(X|X^{(k)}) \) majorize \( F(X) \) and decouple poses from different nodes; see Propositions 3 and 4, respectively. Therefore, we might make use of MM methods where distributed PGO is reduced to independent optimization problems that can be solved in parallel. In Section VII-A, we propose MM methods for distributed PGO using \( G(X|X^{(k)}) \) and \( H(X|X^{(k)}) \). Then, in Section VII-B, we present the algorithm and prove that the proposed method is guaranteed to converge to first-order critical points.

**A. Update Rule**

According to Propositions 3 and 4, \( G(X|X^{(k)}) \) and \( H(X|X^{(k)}) \) are surrogate functions majorizing \( F(X) \)
\[ H(X|X^{(k)}) \geq G(X|X^{(k)}) \geq F(X) \]  \hspace{1cm} (43)
\[ H(X^{(k)}|X^{(k)}) = G(X^{(k)}|X^{(k)}) = F(X^{(k)}). \]  \hspace{1cm} (44)

Following the notion of MM methods [29], we propose the following update rule:
\[ X^{(k+\frac{1}{2})} \leftarrow \arg \min_{X \in \mathcal{A}} H(X|X^{(k)}) \]  \hspace{1cm} (45)
\[ X^{(k+1)} \leftarrow \arg \min_{X \in \mathcal{A}} G(X|X^{(k)}). \]  \hspace{1cm} (46)

Here, \( X^{(k+\frac{1}{2})} \) in (45) is first solved and used to initialize \( X^{(k+1)} \) in (46). Also, (35) and (39) indicate that (45) and (46) are equivalent to \( [\mathcal{A}] \) independent optimization problems of \( X^\alpha \in X^\alpha \) within a single node \( \alpha \)
\[ X^{(k+\frac{1}{2})} \leftarrow \arg \min_{X^\alpha \in \mathcal{A}} H^\alpha(X^\alpha|X^{(k)}) \quad \forall \alpha \in \mathcal{A} \]  \hspace{1cm} (47)
\[ X^{(k+1)} \leftarrow \arg \min_{X^\alpha \in \mathcal{A}} G^\alpha(X^\alpha|X^{(k)}) \quad \forall \alpha \in \mathcal{A} \]  \hspace{1cm} (48)

where \( X^{(k+\frac{1}{2})} \) in (47) is the initial guess to solve \( X^{(k+1)} \) in (48). We remark that (47) and (48) can be solved within a single node \( \alpha \in \mathcal{A} \). Recalling from (42) that \( H^\alpha(X^\alpha|X^{(k)}) = \sum_{i=1}^{n_\alpha} H^i_1(X^{i_1}|X^{(k)}) \), we further reduce (47) to \( n \triangleq \sum_{\alpha \in \mathcal{A}} n_\alpha \) independent optimization problems on a single pose \( X^{i_1} \in \text{SE}(d) \):
\[ X^{(k+\frac{1}{2})} \leftarrow \arg \min_{X^{i_1} \in \text{SE}(d)} H^i_1(X^{i_1}|X^{(k)}) \quad \forall \alpha \in \mathcal{A} \text{ and } i \in \{1, \ldots, n_\alpha \}. \]  \hspace{1cm} (49)

In [43, Appendix K], we have shown that (49) admits an efficient closed-form solution involving only matrix multiplication and singular value decomposition [47].

From (43) and (44), we conclude that (45) to (48) result in
\[ F(X^{(k)}) = H(X^{(k)}|X^{(k)}) \geq H(X^{(k+\frac{1}{2})}|X^{(k)}) \geq F(X^{(k+\frac{1}{2})}) \]  \hspace{1cm} (50a)
\[ F(X^{(k)}) = G(X^{(k)}|X^{(k)}) \geq G(X^{(k+1)}|X^{(k)}) \geq F(X^{(k+1)}) \]  \hspace{1cm} (50b)
which indicate \( F(X^{(k+\frac{1}{2})}) \leq F(X^{(k)}) \) and \( F(X^{(k+1)}) \leq F(X^{(k)}). \) Therefore, (45)–(48) are a reasonable update rule for distributed PGO. In particular, we remark that (45) to (48) combine the strengths of our previous work [31], [32]. Even though (45) and (47) are motivated by Fan and Murphey [31], [32], the proposed method is guaranteed to converge to first-order critical points.
Algorithm 1: The MM--PGO Method.

1: Input: An initial iterate $X^{(0)} \in X$ and $\xi \geq \xi \geq 0$.
2: Output: A sequence of iterates $\{X^{(k)}\}$ and $\{X^{(k+\frac{1}{2})}\}$.
3: for $k \leftarrow 0, 1, 2, \cdots$ do
4: for node $\alpha \leftarrow 1, \ldots, |A|$ do
5: retrieve $X^\beta(k)$ from $\beta \in N_\alpha$.
6: evaluate $\Gamma^\alpha(k), \Pi^\alpha(k), \nabla^\alpha F(X^{(k)})$.
7: $X^{\alpha(k+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in X^\alpha} H^\alpha(X^\alpha | X^{(k)})$ using Algorithm 2
8: $X^{\alpha(k+1)} \leftarrow \arg\min_{X^\alpha \in X^\alpha} G^\alpha(X^\alpha | X^{(k)})$ using
9: with $X^{\alpha(k+\frac{1}{2})}$ as the initial guess.
10: end for
11: end for

Algorithm 2: Solve $X^{\alpha(k+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in X^\alpha} H^\alpha(X^\alpha | X^{(k)})$.

1: Input: $X^\alpha(k), \Pi^\alpha(k), \nabla^\alpha F(X^{(k)})$.
2: Output: $X^{\alpha(k+\frac{1}{2})}$.
3: for $i \leftarrow 1, \ldots, n_\alpha$ do
4: $X^{\alpha(k+\frac{1}{2})}_i \leftarrow \arg\min_{X_i^\alpha \in \mathbb{S}(d)} H^\alpha_i(X^\alpha_i | X^{(k)})$ using
5: $X^{\alpha(k+1)}_i \leftarrow \arg\min_{X_i^\alpha \in X^\alpha} G^\alpha_i(X^\alpha_i | X^{(k)})$ using
6: retrieve $X^{\alpha(k+\frac{1}{2})}$ from $X^{\alpha(k+\frac{1}{2})}_i$ in which $i = 1, \ldots, n_\alpha$.

(46) and (48) make better use of the information within a single node and, thus, take fewer iterations. In contrast to the work in [32], since (45) and (47) have an efficient closed-form solution to (49) that yields sufficient improvement, the time-consuming local minimization of (46) and (48) is avoided as long as $X^{(k+1)}$ and $X^{\alpha(k+1)}$ are initialized with $X^{(k+\frac{1}{2})}$ and $X^{\alpha(k+\frac{1}{2})}$. Most importantly, as shown later in Proposition 5, the proposed update rule of (45–48) has provable convergence to first-order critical points.

B. Algorithm

The proposed update rule results in the MM--PGO method for distributed PGO (see Algorithm 1). The outline of MM--PGO is as follows.

1) In line 5 of Algorithm 1, each node $\alpha$ communicates its information with its neighbors $\beta \in N_\alpha$. Note that no other internode communication is required.
2) In line 6 of Algorithm 1, each node $\alpha$ communicates $\nabla^\alpha F(X^{(k)})$ with $X^\alpha(k)$ and $X^\beta(k)$ where $\beta \in N^\alpha$ are neighbors of node $\alpha$.
3) In line 7 of Algorithm 1, we obtain the intermediate solution $X^{\alpha(k+\frac{1}{2})}$ using Algorithm 2. We have proved that the resulting $X^{\alpha(k+\frac{1}{2})}$ is already sufficient to guarantee the convergence to first-order critical points.
4) In line 4 of Algorithm 2, there exists an exact and efficient closed-form solution to $X^{\alpha(k+\frac{1}{2})}$ using [43, Appendix K].
5) In line 8 of Algorithm 1, we use $X^{\alpha(k+\frac{1}{2})}$ to initialize (48) and improve the final solution $X^{\alpha(k+1)}$ through iterative optimization such that $G^\alpha(X^{\alpha(k+1)} | X^{(k)}) \leq G^\alpha(X^{\alpha(k+\frac{1}{2})} | X^{(k)})$. Note that $X^{\alpha(k+1)}$ does not have to be a local optimal solution to (48); nevertheless, $X^{\alpha(k+\frac{1}{2})}$ is still expected to have a faster convergence than $X^{\alpha(k+\frac{1}{2})}$.

Remark 6: In line 5 of Algorithm 1, node $\alpha$ does not retrieve all the poses in $X^{\beta(k)}$—only poses related to internode measurements are needed and exchanged. This also applies to line 2 of Algorithm 4, and lines 5 and 13 of Algorithm 5 in the following sections.

Remark 7: MM--PGO (see Algorithm 1) requires no internode communication except for line 5 of Algorithm 1 that is used to evaluate $\Gamma^\alpha(k), \Pi^\alpha(k), \nabla^\alpha F(X^{(k)})$, which, as mentioned before, can be distributed with one internode communication round between neighboring nodes $\alpha$ and $\beta$ without introducing any additional computation.

Since $X^{\alpha(k+\frac{1}{2})}$ in (47) has a closed-form solution that can be efficiently computed, and (48) does not require $X^{\alpha(k+1)}$ to be a local optimal solution, the overall computational efficiency of the MM--PGO method is significantly improved in contrast to the work in [31] and [32]. More importantly, the MM--PGO method still converges to first-order critical points as long as the following assumption holds.

Assumption 3: For $X^{\alpha(k+1)}$ and $X^{\alpha(k+\frac{1}{2})}$, it is assumed that

$$G^\alpha(X^{\alpha(k+1)} | X^{(k)}) \leq G^\alpha(X^{\alpha(k+\frac{1}{2})} | X^{(k)})$$

for each node $\alpha = 1, 2, \ldots, |A|$.

Note that Assumption 3 can be satisfied with ease as long as line 8 of Algorithm 1 is initialized with $X^{\alpha(k+\frac{1}{2})}$. Then, we have the following proposition about the convergence of MM--PGO (see Algorithm 1).

Proposition 5: If Assumptions 1 to 3 hold, then for a sequence of $\{X^{(k)}\}$ and $\{X^{(k+\frac{1}{2})}\}$ generated by Algorithm 1, we have the following.

1) $F(X^{(k)})$ is nonincreasing as $k \to \infty$.
2) $F(X^{(k)}) \to F^\infty$ as $k \to \infty$.
3) $|X^{(k+1)} - X^{(k)}| \to 0$ as $k \to \infty$ if $\xi > 0$.
4) $|X^{(k+\frac{1}{2})} - X^{(k)}| \to 0$ as $k \to \infty$ if $\zeta > \xi > 0$.
5) If $\zeta > \xi > 0$, then there exists $\epsilon > 0$ such that

$$\min_{0 \leq \zeta < K} |\nabla F(X^{(k+\frac{1}{2})})| \leq \sqrt{\frac{2}{\epsilon}} \cdot \frac{F(X^{(0)}) - F^\infty}{K + 1}$$

for any $K \geq 0$.
6) If $\zeta > \xi > 0$, then $\nabla F(X^{(k)}) \to 0$ and $\nabla F(X^{(k+\frac{1}{2})}) \to 0$ as $k \to \infty$.

Proof: See [43, Appendix E].

Remark 8: In contrast to other distributed PGO algorithms [36], [37], [38], [39], MM--PGO has the mildest conditions for convergence and applies to a broad class of loss kernels.

VIII. ACCELERATED MM METHOD FOR DISTRIBUTED PGO WITH MASTER NODE

In the last several decades, a number of accelerated first-order optimization methods have been proposed [33], [34]. Even though most of them were originally developed for convex
optimization, it has been found that these accelerated methods also work well for nonconvex optimization [48], [49], [50]. In our previous works [31], [32], we proposed to use Nesterov’s method to accelerate distributed PGO, which yields much faster convergence. Since MM–PGO is a first-order optimization method, it is possible to exploit Nesterov’s method for acceleration. In Section VIII-A, we implement Nesterov’s method to accelerate MM methods for distributed PGO. Then, in Section VIII-B, we introduce the adaptive restart scheme [35] to guarantee the convergence if a master node exists. Finally, in Section VIII-C, we propose the accelerated MM method for distributed PGO with master and prove that such a method converges to first-order critical points.

A. Nesterov’s Method

According to Propositions 3 and 4, (45) and (46) are proximal operators of $F(X)$, making it possible to implement Nesterov’s method [33], [34] for acceleration and resulting in the following update rule for $X^{\alpha(k+\frac{1}{2})}$ and $X^{\alpha(k+1)}$:

\begin{align}
    s^{\alpha(k+1)} &= \frac{\sqrt{4s^{\alpha(k)}+1}+1}{2} \\
    \lambda^{\alpha(k)} &= \frac{s^{\alpha(k)}-1}{s^{\alpha(k)}+1} \\
    Y^{\alpha(k)} &= X^{\alpha(k)} + \lambda^{\alpha(k)} \cdot \left(X^{\alpha(k)} - X^{\alpha(k-1)}\right) \\
    X^{\alpha(k+\frac{1}{2})} &= \underset{X^{\alpha} \in \mathcal{X}^{\alpha}}{\arg\min} H^{\alpha}(X^{\alpha})Y^{\alpha(k)} \\
    X^{\alpha(k+1)} &= \underset{X^{\alpha} \in \mathcal{X}^{\alpha}}{\arg\min} G^{\alpha}(X^{\alpha})Y^{\alpha(k)}. \tag{56}
\end{align}

In (55) and (56), $G^{\alpha}(-|Y^{\alpha(k)}) : \mathcal{X}^{\alpha} \rightarrow \mathbb{R}$ and $H^{\alpha}(-|Y^{\alpha(k)}): \mathcal{X}^{\alpha} \rightarrow \mathbb{R}$ are surrogate functions at $Y^{\alpha(k)}$:

\begin{align}
    G^{\alpha}(X^{\alpha}|Y^{\alpha(k)}) &= \frac{1}{2}\|X^{\alpha} - Y^{\alpha(k)}\|_{\Pi^{\alpha(k)}}^2 + \langle \nabla X^{\alpha} F(Y^{\alpha(k)}), X^{\alpha} - Y^{\alpha(k)} \rangle \tag{57} \\
    H^{\alpha}(X^{\alpha}|Y^{\alpha(k)}) &= \frac{1}{2}\|X^{\alpha} - Y^{\alpha(k)}\|_{\Pi^{\alpha(k)}}^2 + \langle \nabla X^{\alpha} F(Y^{\alpha(k)}), X^{\alpha} - Y^{\alpha(k)} \rangle \tag{58}
\end{align}

where $\Pi^{\alpha(k)}$ and $\Pi^{\alpha(k)}$ are the same as these in $G^{\alpha}(-|X^{\alpha(k)})$ and $H^{\alpha}(-|X^{\alpha(k)})$ in (36) and (40).

The key insight of Nesterov’s method is to exploit the momentum $X^{\alpha(k)} - X^{\alpha(k-1)}$ for acceleration, which is essentially governed by (52)–(54). Note that Nesterov’s method using (52)–(56) is equivalent to (47) to (48) when $s^{\alpha(k)} = 1$ and $\lambda^{\alpha(k)} = 0$, and then increasingly affected by the momentum as $s^{\alpha(k)}$ and $\lambda^{\alpha(k)}$ increase.

Nesterov’s method is known to converge quadratically for convex optimization while the unaccelerated MM method only has linear convergence [33], [34]. Even though distributed PGO is a nonconvex optimization problem, similar to the work in [31] and [32], (52)–(56) using Nesterov’s method for acceleration empirically have significant speedup while introducing almost no extra computation or communication compared to the MM–PGO method. Thus, it is preferable to adopt Nesterov’s method to accelerate distributed PGO.

B. Adaptive Restart

In spite of faster convergence, Nesterov’s accelerated distributed PGO using (52) to (56) is no longer nonincreasing and might fail to converge due to the nonconvexity of PGO. Fortunately, such a problem can be remedied with an adaptive restart scheme [31], [32], [35] as the following.

Let $\overline{F}^{(k)}$ be an exponential moving averaging of $F(X^{(0)})$, $F(X^{(1)})$, $\ldots$, $F(X^{(k)})$:

\begin{align}
    \overline{F}^{(k)}(k) &= \begin{cases} 
    F(X^{(0)}), & k = 0 \\
    (1 - \eta) \cdot \overline{F}^{(k-1)} + \eta \cdot F(X^{(k)}), & \text{otherwise}
    \end{cases} \tag{59}
\end{align}

where $\eta \in (0, 1]$. Following the work in [31], [50], and [51], the adaptive restart scheme guarantees the convergence by keeping $F(X^{(k+1)}) \leq \overline{F}^{(k)}$. Even though it is not obvious, $F(X^{(k+1)}) \leq \overline{F}^{(k)}$ can be achieved with the following steps.

1) Update $X^{(k+\frac{1}{2})}$ and $X^{(k+1)}$ by solving (55) and (56) for each node $\alpha \in \mathcal{A}$.
2) If $F(X^{(k+\frac{1}{2})}) > \overline{F}^{(k)}$, update $X^{(k+\frac{1}{2})}$ again by solving (47) for each node $\alpha \in \mathcal{A}$.
3) If $F(X^{(k+1)}) > \overline{F}^{(k)}$, update $X^{(k+1)}$ again by solving (48) and reduce $\alpha^{\text{iter}}(k+1)$ for each node $\alpha \in \mathcal{A}$.

Due to space limitation, the complete analysis of the adaptive restart scheme is left in [43, Appendix F] where more details are presented.

Remark 9: Since $\eta \in (0, 1]$ in (59), then $\overline{F}^{(k+1)} \leq F^{(k+1)}$ as long as $F(X^{(k+1)}) \leq \overline{F}^{(k)}$. Note that (47) and (48) lead to $F(X^{(k+\frac{1}{2})}) \leq F(X^{(k)})$ and $F(X^{(k+1)}) \leq F(X^{(k)})$, and we have proved in [43, Appendix F] that $F(X^{(k)}) \leq \overline{F}^{(k)}$. This suggests $F(X^{(k+1)}) \leq \overline{F}^{(k)}$ if $X^{(k+1)}$ is updated from (48), and thus, satisfies the restart conditions. Then, the adaptive restart scheme above results in a nonincreasing sequence of $\overline{F}^{(k)}$. Furthermore, [43, Appendix F] indicates that such an adaptive restart scheme is sufficient to guarantee the convergence to first-order critical points under mild conditions.

Note that one has to aggregate information across the network to evaluate and compare $\overline{F}^{(k)}$, $F(X^{(k+\frac{1}{2})})$, $F(X^{(k+1)})$ using (18) and (59). Thus, a master node capable of communicating with each node $\alpha \in \mathcal{A}$ is required. In the rest of this section, we make the following assumption about the existence of such a master node.

Assumption 4: There is a master node to retrieve $X^{\alpha(k)}$ and $X^{\alpha(k+\frac{1}{2})}$ from each node $\alpha \in \mathcal{A}$ and evaluate $\overline{F}^{(k)}$, $F(X^{(k+\frac{1}{2})})$, $F(X^{(k+1)})$.

C. Algorithm

Implementing Nesterov’s method and the adaptive restart scheme, we obtain the AMM–PGO* method for distributed PGO (Algorithm 3), where “∗” indicates the existence of a master node.

The outline of AMM–PGO* is as follows.
Algorithm 3: The AMM–PGO* Method.

1: **Input:** An initial iterate $X^{(0)} \in X$, and $\zeta \geq 0$, and $\eta \in (0, 1)$, and $\psi > 0$, and $\phi > 0$.
2: **Output:** A sequence of iterates $\{X^{(k)}\}$ and $\{X^{(k+\frac{1}{2})}\}$.
3: **for** node $\alpha \leftarrow 1, \ldots, |A|$ **do**
4: $X^{(\alpha-1)} \leftarrow X^{(\alpha)(0)}$ and $s^{\alpha}(0) \leftarrow 1$
5: send $X^{(\alpha)(0)}$ to the master node
6: **end for**
7: evaluate $F(X^{(0)})$ using (17) at the master node
8: $F^{(-1)} \leftarrow F(X^{(\alpha)})$ at the master node
9: **for** $k \leftarrow 0, 1, \ldots$ **do**
10: **for** node $\alpha \leftarrow 1, \ldots, |A|$ **do**
11: $s^{\alpha}(k+1) \leftarrow \frac{\sqrt{2}}{\alpha^{\alpha(k-1)+\frac{1}{2}}} + \frac{\alpha^{(k)}}{\alpha^{(k)}}$
12: $Y^{(k)} \leftarrow X^{(\alpha)(k)} + \frac{\alpha^{(k)}}{\alpha^{(k)}} \cdot (X^{(\alpha)} - X^{(\alpha)(k-1)})$
13: **end for**
14: $F^{(k)} \leftarrow (1 - \eta) \cdot F^{(k-1)} + \eta \cdot F(X^{(k)})$ at the master node
15: update $X^{(k+\frac{1}{2})}$ and $X^{(k+1)}$ using Algorithm 4
16: **end for**

1) In lines 11 and 12 of Algorithm 3, each node $\alpha$ computes $Y^{(k)}$ for Nesterov’s acceleration that is related to $s^{\alpha}(k) \in [1, \infty)$ and $\alpha^{(k)} \in [0, 1]$.
2) In line 2 of Algorithm 4, each node $\alpha$ performs one inter-node communication round to retrieve $Y^{(\beta)}(k)$ and $Y^{(\beta)(k)}$ from its neighbors $\beta \in \mathcal{N}^{\alpha}$.
3) In line 5 of Algorithm 3 and lines 6, 12, and 20 of Algorithm 4, each node $\alpha$ performs one inter-node communication round to send $X^{(\alpha)(k+\frac{1}{2})}$ and $X^{(\alpha)(k+1)}$ to the master node.
4) In line 3 of Algorithm 4, each node $\alpha$ evaluates $\Gamma^{(k)} \Pi^{(\alpha)}$, $\nabla X^{\alpha}(F(X^{(k)}))$, and $\nabla X^{\alpha}(F(Y^{(1)}))$ using $X^{(\alpha)(k)}$, $Y^{(\alpha)}(k)$, and $Y^{(\alpha)(k)}$ where $\beta \in \mathcal{N}^{\alpha}$ are neighbors of node $\alpha$.
5) In lines 8 and 14 of Algorithm 3 and lines 8, 14, and 22 of Algorithm 4, the master node evaluates $F^{(k)}$, $F(X^{(k+\frac{1}{2})})$, and $F(X^{(k+1)})$ that are used for adaptive restart.
6) In lines 9–23 of Algorithm 4, the master node performs an adaptive restart to keep $F(X^{(k+\frac{1}{2})}) \leq F^{(k)}$ and $F(X^{(k+1)}) \leq F^{(k)}$, which yields a nonincreasing sequence of $F^{(k)}$ to guarantee the convergence.
7) In lines 5 and 18 of Algorithm 4, note that $X^{(\alpha)(k+1)}$ does not have to be a local optimal solution to (48).
8) In lines 24–26 of Algorithm 4, $F(X^{(k+1)})$ is guaranteed to yield sufficient improvement over $F^{(k)}$ in comparison to $F(X^{(k+\frac{1}{2})})$.

In spite of acceleration, AMM–PGO converges to first-order critical points as the following proposition states.

**Proposition 6:** If Assumption 1–4 hold, $\psi > 0$ and $\phi > 0$, then for a sequence of $\{X^{(k)}\}$ and $\{X^{(k+\frac{1}{2})}\}$ generated by Algorithm 3, we have the following.
1) $F^{(k)}$ is nonincreasing.
2) $F(X^{(k)}) \rightarrow F^{\infty}$ and $F^{(k)} \rightarrow F^{\infty}$ as $k \rightarrow \infty$.

Algorithm 4: Updates for the AMM–PGO* Method.

1: **for** node $\alpha \leftarrow 1, \ldots, |A|$ **do**
2: retrieve $X^{(\beta)}(k)$ and $Y^{(\beta)}(k)$ from $\beta \in \mathcal{N}_{\alpha}$
3: evaluate $\Gamma^{(\alpha)}(k), \Pi^{(\alpha)}(k), \nabla X^{\alpha}(F(X^{(k)})), \nabla X^{\alpha}(F(Y^{(k)}))$
4: $X^{(\alpha)(k+\frac{1}{2})} \leftarrow \arg\min_{X^{\alpha} \in \mathcal{X}^{\alpha}} H^{\alpha}(X^{\alpha})|Y^{(k)}|$ using Algorithm 2
5: $X^{(\alpha)(k+1)} \leftarrow \arg\min_{X^{\alpha} \in \mathcal{X}^{\alpha}} G^{\alpha}(X^{\alpha} | Y^{(k)})$ with $X^{(\alpha)(k+\frac{1}{2})}$ as the initial guess
6: send $X^{(\alpha)(k+\frac{1}{2})}$ and $X^{(\alpha)(k+1)}$ to the master node
7: **end for**
8: evaluate $F(X^{(k+\frac{1}{2})})$ and $F(X^{(k+1)})$ using (17) at the master node
9: **if** $F(X^{(k+\frac{1}{2})}) > F^{(k)} - \psi \cdot \|X^{(k+\frac{1}{2})} - X^{(k)}\|^2$ **then**
10: **for** node $\alpha \leftarrow 1, \ldots, |A|$ **do**
11: $X^{(\alpha)(k+1)} \leftarrow \arg\min_{X^{\alpha} \in \mathcal{X}^{\alpha}} H^{\alpha}(X^{\alpha} | X^{(k)})$ using Algorithm 2
12: send $X^{(\alpha)(k+1)}$ to the master node
13: **end for**
14: evaluate $F(X^{(k+\frac{1}{2})})$ using (17) at the master node
15: **end if**
16: **if** $F(X^{(k+1)}) > F^{(k)} - \psi \cdot \|X^{(k+1)} - X^{(k)}\|^2$ **then**
17: **for** node $\alpha \leftarrow 1, \ldots, |A|$ **do**
18: $X^{(\alpha)(k+1)} \leftarrow \arg\min_{X^{\alpha} \in \mathcal{X}^{\alpha}} G^{\alpha}(X^{\alpha} | X^{(k)})$
19: $s^{\alpha}(k+1) \leftarrow \max\{\frac{1}{2}, s^{(k+1)}\}$
20: send $X^{(\alpha)(k+1)}$ to the master node
21: **end for**
22: evaluate $F(X^{(k+1)})$ using (17) at the master node
23: **end if**
24: **if** $F^{(k)} - F(X^{(k+1)}) < \phi \cdot (F^{(k)} - F(X^{(k+\frac{1}{2})}))$ **then**
25: $X^{(k+1)} \leftarrow X^{(k+\frac{1}{2})}$ and $F(X^{(k+1)}) \leftarrow F(X^{(k+\frac{1}{2})})$
26: **end if**

3) $\|X^{(k+1)} - X^{(k)}\| \rightarrow 0$ as $k \rightarrow \infty$ if $\xi > 0$ and $\zeta > 0$. 
4) $\|X^{(k+\frac{1}{2})} - X^{(k)}\| \rightarrow 0$ as $k \rightarrow \infty$ if $\xi \leq \zeta > 0$.
5) If $\zeta \geq 0$, then there exists $\epsilon > 0$ such that
$$\min_{0 \leq k < K} \|\grad F(X^{(k+\frac{1}{2})})\| \leq \frac{1}{\epsilon} \cdot \frac{F(X^{(0)}) - F^{\infty}}{K + 1}$$
for any $K \geq 0$.
6) If $\zeta > \xi > 0$, then $\grad F(X^{(k)}) \rightarrow 0$ and $\grad F(X^{(k+\frac{1}{2})}) \rightarrow 0$ as $k \rightarrow \infty$.

**Proof:** See [43, Appendix F].

**Remark 10:** If $\eta = 1$ in (59), $F(X^{(k)}) = F^{(k)}$, and $F(X^{(k)})$ is also nonincreasing according to Proposition 6(a). While $F(X^{(k)})$ might fail to be nonincreasing, we still recommend to choose $\eta \ll 1$ that empirically yields fewer adaptive restarts and faster convergence for distributed PGO.

**Remark 11:** If $\psi > 0$ and $\phi > 0$ in Algorithm 4 guarantee that $F(X^{(k+\frac{1}{2})})$ and $F(X^{(k+1)})$ yield sufficient improvement over $F^{(k)}$ in terms of $\|X^{(k+\frac{1}{2})} - X^{(k)}\|$ and $\|X^{(k+1)} - X^{(k)}\|$ and are recommended to set close to zero to avoid unnecessary adaptive restarts and make full use of Nesterov’s acceleration.
IX. ACCELERATED MM METHOD FOR DISTRIBUTED PGO WITHOUT MASTER NODE

The adaptive restart is essential for the convergence of accelerated MM methods. In AMM–PGO* (Algorithm 3), the adaptive restart scheme needs a master node to evaluate \( F(X^{(k+1)}) \) and \( \overline{F}^{(k)} \) and guarantee the convergence. On the other hand, if there is no master node, the adaptive restart scheme requires a substantial amount of internode communication, making AMM–PGO* unsalable for large-scale distributed PGO. Recently, we developed an adaptive restart scheme for distributed PGO that does not require a master node while generating convergent iterates [32]. Nevertheless, the adaptive restart scheme in [32] is conservative and suffers from unnecessary restarts that hinder acceleration and yield slower convergence. Thus, we need to redesign the adaptive restart scheme for distributed PGO without a master node to maximize the performance of accelerated MM methods.

To address this issue, in Section IX-A, we develop a novel adaptive restart scheme that requires no master node and is fully decentralized. Then, in Section IX-B, we propose the accelerated MM method for distributed PGO without master that has provable convergence to first-order critical points. In particular, the resulting accelerated MM method, which needs no master node and is fully decentralized, empirically has no loss of computational efficiency in contrast to AMM–PGO* with a master node; see Section X for more details.

A. Adaptive Restart

Recall that AMM–PGO*’s adaptive restart scheme guarantees the convergence by keeping \( F(X^{(k+1)}) \leq \overline{F}^{(k)} \), where the master node only evaluates and compares \( F(X^{(k+1)}) \) and \( \overline{F}^{(k)} \). This suggests that if we could achieve \( F(X^{(k+1)}) \leq \overline{F}^{(k)} \) without evaluating and comparing \( F(X^{(k+1)}) \) and \( \overline{F}^{(k)} \), no master node will be needed. We also note that if there is a sequence of \( \{F^{(k)}\} \) and \( \{\overline{F}^{(k)}\} \) for each node \( \alpha \) such that

\[
F(X^{(k)}) = \sum_{\alpha \in A} F^{(k)}
\]

\[
\overline{F}^{(k)} = \sum_{\alpha \in A} \overline{F}^{(k)}
\]

\[
F^{(k+1)} \leq \overline{F}^{(k)}
\]

then \( F(X^{(k+1)}) = \sum_{\alpha \in A} F^{(k+1)} \leq \sum_{\alpha \in A} \overline{F}^{(k)} = \overline{F}^{(k)} \).

Therefore, the sequence above of \( \{F^{(k)}\} \) and \( \{\overline{F}^{(k)}\} \) is sufficient to keep \( F(X^{(k+1)}) \leq \overline{F}^{(k)} \) despite that \( F(X^{(k+1)}) \) and \( \overline{F}^{(k)} \) are not explicitly evaluated and compared. More importantly, an adaptive restart scheme without a master node can be developed with the sequence. In the rest of this section, we will construct \( \{F^{(k)}\} \) and \( \{\overline{F}^{(k)}\} \) satisfying (60)–(62), which further results in the adaptive restart scheme for distributed PGO without a master node.

For notational simplicity, we define \( \Delta G^{(k)}(X|X^{(k)}) : X \to \mathbb{R} \) related to the majorization gap of \( G(X|X^{(k)}) \) over \( F(X) \)

\[
\Delta G^{(k)}(X|X^{(k)}) \triangleq \frac{\ell}{2} \|X^{(k)} - X^{(k)}\|^2
\]

\[
+ \frac{1}{2} \sum_{\beta \in N^\alpha} \sum_{(i,j) \in E^\alpha} \left( F^{ij\beta}(X) - E^{ij\beta}(X|X^{(k)}) \right)
\]

\[
+ \frac{1}{2} \sum_{\beta \in N^\alpha} \sum_{(i,j) \in E^\alpha} \left( F^{ij\beta}(X) - E^{ij\beta}(X|X^{(k)}) \right)
\]

(63)

where \( F^{ij\beta}(X), F^{ij\beta}(X), E^{ij\beta}(X|X^{(k)}), \) and \( E^{ij\beta}(X|X^{(k)}) \) are given in (19) and (31). From \( \Delta G^{(k)}(X|X^{(k)}) \) in (63), we recursively define \( F^{(k)}, \overline{F}^{(k)}, \) and \( G^{(k)} \) according to the following.

1) If \( k = -1 \), each node \( \alpha \) initializes \( F^{(0)} \) and \( \overline{F}^{(0)} \) with

\[
F^{(0)} = \sum_{(i,j) \in E^\alpha} F^{ij}(X^{(0)})
\]

\[
\overline{F}^{(0)} = \sum_{(i,j) \in E^\alpha} \overline{F}^{ij}(X^{(0)})
\]

\[
+ \frac{1}{2} \sum_{\beta \in N^\alpha} \sum_{(i,j) \in E^\alpha} \left( F^{ij\beta}(X^{(0)}) - E^{ij\beta}(X^{(0)}) \right)
\]

\[
+ \frac{1}{2} \sum_{\beta \in N^\alpha} \sum_{(i,j) \in E^\alpha} \left( F^{ij\beta}(X^{(0)}) - E^{ij\beta}(X^{(0)}) \right)
\]

(64)

\[
F^{(0)} = F^{(0)}
\]

(65)

2) If \( k \geq 0 \), each node \( \alpha \) recursively updates \( G^{(k)}(X|X^{(k)}) \) and \( \overline{F}^{(k)} \) according to

\[
G^{(k)}(X|X^{(k)}) \triangleq G^{(k)}(X^{(k)}|X^{(k-1)}) + F^{(k-1)}(X^{(k-1)})
\]

\[
F^{(k)} = G^{(k)} + \Delta G^{(k)}(X^{(k)}|X^{(k-1)})
\]

(66)

\[
\overline{F}^{(k)} = (1 - \eta) \cdot F^{(k-1)} + \eta \cdot G^{(k)}
\]

(67)

(68)

where \( \eta \in (0, 1] \).

In [43, Appendix G], we have proved that such a sequence of \( \{F^{(k)}\} \) and \( \{\overline{F}^{(k)}\} \) satisfies (60)–(62) as long as \( G^{(k)} \leq \overline{F}^{(k)} \), which yields the following proposition.

**Proposition 7:** For \( G^{(k)}(X|X^{(k)}) \), \( F^{(k)} \), and \( \overline{F}^{(k)} \) in (64) to (68), we have the following.

1) \( F(X^{(k)}) = \sum_{\alpha \in A} F^{(k)} \) where \( F(X^{(k)}) \) is given in (17).
2) \( \overline{F}^{(k)} = \sum_{\alpha \in A} \overline{F}^{(k)} \) where \( \overline{F}^{(k)} \) is given in (59).
3) \( F^{(k)} \leq \overline{F}^{(k)} \leq \overline{F}^{(k)} \) if \( G^{(k+1)} \leq \overline{F}^{(k)} \).

**Proof:** See [43, Appendix G].

It can be concluded from Propositions 7(a) and 7(b) that the resulting \( \{F^{(k)}\} \) and \( \{\overline{F}^{(k)}\} \) satisfies (60) and (61), and Proposition 7(c) indicates that (62) holds if \( G^{(k+1)} \leq \overline{F}^{(k)} \).

In [43, Appendix H], we have also proved that the following steps are sufficient to lead to \( G^{(k+1)}(X|X^{(k)}) \leq \overline{F}^{(k)} \):

1) Update \( X^{(k+1)} \) by solving (56) at node \( \alpha \).
2) Compute \( G^{(k+1)}(X|X^{(k)}) \) at node \( \alpha \).
3) If \( G^{(k+1)}(X|X^{(k)}) > \overline{F}^{(k+1)} \), update \( X^{(k+1)} \) again by solving (48) and reduce \( \theta^{(k+1)} \) at node \( \alpha \).
Algorithm 5: The AMM–PGO\# Method.

1: Input: An initial iterate $X^{(0)} \in \mathcal{X}$, and $\eta \in (0, 1]$, and $\zeta \geq \xi \geq 0$, and $\psi > 0$, and $\phi > 0$.
2: Output: A sequence of iterates $\{X^{(k)}\}$ and $\{X^{(k+\frac{1}{2})}\}$.
3: for node $\alpha \leftarrow 1, \ldots, |\mathcal{A}|$ do
4: $X^{(\alpha)} \leftarrow X^{(0)}$ and $s^{(\alpha)} \leftarrow 1$
5: retrieve $X^{(\beta)}(1)$ and $X^{(\beta)}(0)$ from $\beta \in N_{\alpha}$
6: evaluate $F^{(\alpha)}(1)$, $F^{(\alpha)}(0)$ using (64) and (65)
7: $G^{(\alpha)}(0) \leftarrow G^{\alpha}(X^{(\alpha)}(1)) + F^{(\alpha)}(1)$
8: end for
9: for $k \leftarrow 0, 1, 2, \ldots$ do
10: for node $\alpha \leftarrow 1, \ldots, |\mathcal{A}|$ do
11: $s^{(\alpha)}(k+1) \leftarrow s^{(\alpha)}(k) + 1$
12: $Y^{(\alpha)}(k) \leftarrow X^{(\alpha)} + \lambda^{(\alpha)}(k), (X^{(\alpha)} - X^{(\alpha)(k-1)})$
13: retrieve $X^{(\beta)}(k)$ and $Y^{(\beta)}(k)$ from $\beta \in N_{\alpha}$
14: $F^{(\alpha)}(k) \leftarrow G^{(\alpha)}(k) + \Delta G^{\alpha}(X^{(\alpha)}(k)|X^{(\alpha)})$ using (63) and (66)
15: $F^{(\alpha)}(k) \leftarrow (1 - \eta) \cdot F^{(\alpha)}(k-1) + \eta \cdot F^{(\alpha)}(k)$
16: update $X^{(\alpha)(k+\frac{1}{2})}$ and $X^{(\alpha)(k+1)}$ using Algorithm 6
17: end for
18: end for

Then, we not only obtain a sequence of $\{F^{(\alpha)}(k)\}$ and $\{F^{(\alpha)}(k)\}$ satisfying (60)–(62), but also an adaptive restart scheme using $G^{(\alpha)}(k), F^{(\alpha)}(k),$ and $F^{(\alpha)}(k)$ to keep $F(X^{(k+1)}) \leq F^{(\alpha)}(k)$. Note that $F(X^{(k+1)})$ and $F^{(\alpha)}(k)$ are neither evaluated nor compared. Instead, we evaluate and compare $G^{(\alpha)}(k+1)$ and $F^{(\alpha)}(k)$ independently at each node $\alpha$. Moreover, according to (19), (36), and (63), it is tedious but straightforward to show that $G^{(\alpha)}(k), F^{(\alpha)}(k),$ and $F^{(\alpha)}(k)$ in (64)–(68) can be computed with one internode communication round between node $\alpha$ and its neighbors $\beta \in N_{\alpha}$. We emphasize that such an adaptive restart scheme differs from those in AMM–PGO* and [31], [50], [51] that have a master node to evaluate and compare $F(X^{(k+1)})$ and $F^{(\alpha)}(k)$. In contrast, the resulting adaptive restart scheme keeps $F(X^{(k+1)}) \leq F^{(\alpha)}(k)$ but needs no master node and, thus, is well-suited for distributed PGO without a master node.

Remark 12: $F^{(\alpha)}(X) - E^{(\alpha)}(X|X^{(k)})$ and $F^{(\alpha)}(X) - E^{(\alpha)}(X|X^{(k)})$ are majorization gaps of internode measurements related to nodes $\alpha$ and $\beta$. According to (63), $\Delta G^{\alpha}(X^{(\alpha)}(k)|X^{(\alpha)})$ takes half of these majorization gaps of internode measurements for node $\alpha$. Then, (67) uses $\Delta G^{\alpha}(X^{(\alpha)}|X^{(\alpha)})$ to compute $F^{(\alpha)}(k)$ with majorization gaps $E^{(\alpha)}(X|X^{(k)})$ and $F^{(\alpha)}(X) - E^{(\alpha)}(X|X^{(k)})$ even redistributed between nodes $\alpha$ and $\beta$. However, $F^{(\alpha)}(X)$ and $F^{(\alpha)}(X)$ of internode measurements might fail to be evenly distributed between nodes $\alpha$ and $\beta$ for $k > 0$.

B. Algorithm

With the adaptive restart scheme using $G^{(\alpha)}(k), F^{(\alpha)}(k),$ and $F^{(\alpha)}(k)$ to keep $F(X^{(k+1)}) \leq F^{(\alpha)}(k)$, we obtain the AMM–PGO# method (see Algorithm 5) for distributed PGO, where “#” indicates that no master node is needed.

The following outline of AMM–PGO# is similar to AMM–PGO* and the key difference is the adaptive restart scheme.

1) In lines 5 and 13 of Algorithm 5, each node $\alpha$ performs one internode communication round to retrieve $X^{(\alpha)}(k)$ and $Y^{(\alpha)}(k)$ from its neighbors $\beta \in N_{\alpha}$. We remark that no other internode communication is required.
2) In lines 6, 14, and 15 of Algorithm 5 and lines 4, 6, 9, and 13 of Algorithm 6, each node $\alpha$ evaluates $F^{(\alpha)}(k), F^{(\alpha)}(k),$ and $G^{(\alpha)}(k+\frac{1}{2})$ that are used for adaptive restart. Note that $X^{(\alpha)}$ and $X^{(\alpha)}$ from node $\alpha$’s neighbors $\beta \in N_{\alpha}$ are needed.
3) In lines 7–15 of Algorithm 6, each node $\alpha$ performs an independent adaptive restart such that $G^{(\alpha)}(k+\frac{1}{2}) \leq F^{(\alpha)}(k)$ and $G^{(\alpha)}(k+1) \leq F^{(\alpha)}(k)$, which also results in $F(X^{(k+1)}) \leq F^{(\alpha)}(k)$ and a nonincreasing sequence of $F^{(\alpha)}(k)$ for distributed PGO without a master node.
4) In lines 16–18 of Algorithm 6, $G^{(\alpha)}(k+1)$ is guaranteed to yield sufficient improvement over $F^{(\alpha)}(k)$ compared to $G^{(\alpha)(k+\frac{1}{2})}$.

Furthermore, AMM–PGO# converges to first-order critical points as the following propositions states.

Proposition 8: If Assumption 1–3 hold, $\psi > 0$ and $\phi > 0$, then for a sequence of $\{X^{(k)}\}$ and $\{X^{(k+\frac{1}{2})}\}$ generated by Algorithm 5, we have the following.

1) $F^{(k)}$ is nonincreasing.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
with 5 robots despite that the later has a much faster convergence if

\[ F(X^{(k)}) \to F^\infty \quad \text{and} \quad F^{(k)} \to F^\infty \quad \text{as} \quad k \to \infty. \]

3) \[ \|X^{(k+1)} - X^{(k)}\| \to 0 \quad \text{as} \quad k \to \infty \quad \text{if} \quad \zeta > \xi > 0. \]

4) \[ \|X^{(k+\frac{1}{2})} - X^{(k)}\| \to 0 \quad \text{as} \quad k \to \infty \quad \text{if} \quad \zeta > \xi > 0. \]

5) If \( \zeta \geq \xi > 0 \), then there exists \( \epsilon > 0 \) such that

\[ \min_{0 \leq k < K} \| \text{grad} F(X^{(k+\frac{1}{2})}) \| \leq 2 \frac{1}{\epsilon} \frac{F(X^{(0)}) - F^\infty}{K + 1} \]

for any \( K \geq 0 \).

6) If \( \zeta \geq \xi > 0 \), then \( \text{grad} F(X^{(k)}) \to 0 \) and \( \text{grad} F(X^{(k+\frac{1}{2})}) \to 0 \) as \( k \to \infty \).

**Proof:** See [43, Appendix H]. □

In spite of no master node, Proposition 8 indicates that AMM–PGO# has provable convergence as long as each node \( \alpha \in \mathcal{A} \) can communicate with its neighbors \( \beta \in \mathcal{N}^\alpha \). Thus, AMM–PGO# eliminates the bottleneck of communication for distributed PGO without a master node. In addition, AMM–PGO# also reduces unnecessary adaptive restarts compared to that in [32] and, thus, makes better of Nesterov’s acceleration and has faster convergence.

**X. EXPERIMENTS**

In this section, we evaluate the performance of our MM methods (MM–PGO, AMM–PGO*, and AMM–PGO#) for distributed PGO on the simulated Cube datasets and a number of 2-D and 3-D SLAM benchmark datasets [8]. In terms of MM–PGO, AMM–PGO*, and AMM–PGO#, \( \eta, \zeta, \psi, \) and \( \phi \) in Algorithms 1, 3, and 5 are 5 \( \times \) 10\(^{-4}\), 1 \( \times \) 10\(^{-10}\), 1.5 \( \times \) 10\(^{-10}\), 1 \( \times \) 10\(^{-10}\), and 1 \( \times \) 10\(^{-6}\), respectively, for all the experiments. In addition, MM–PGO, AMM–PGO*, and AMM–PGO# can take at most one iteration when solving (48) and (56) to improve the estimates. All the experiments have been performed on a laptop with an Intel Xeon(R) CPU E3-1535M v6 and 64 GB of RAM running Ubuntu 20.04.

**A. Cube Datasets**

In this section, we test and evaluate our MM methods for distributed PGO on 20 simulated Cube datasets (see Fig. 2) with 5, 10, and 50 robots. In the experiment, a simulated Cube dataset has 12 \( \times \) 12 \( \times \) 12 cube grids with 1 m side length, a path of 3600 poses along the rectilinear edge of the cube grid, odometric measurements between all the pairs of sequential poses, and loop-closure measurements between nearby but nonsequential poses that are randomly available with a probability of 0.1. We generate the odometric and loop-closure measurements according to the noise models in [8] with an expected translational RMSE of 0.02 m and an expected angular RMSE of 0.02\( \pi \) rad. The centralized chordal initialization [40] is implemented such that distributed PGO with different numbers of robots have the same initial estimate. The maximum number of iterations is 1000.

We evaluate the convergence of MM–PGO, AMM–PGO*, and AMM–PGO# in terms of the relative suboptimality gap and Riemannian gradient norm. For reference, we also make comparisons against AMM–PGO [32]. Note that AMM–PGO is the original accelerated MM method for distributed PGO whose adaptive restart scheme is conservative and might prohibit Nesterov’s acceleration.

**Relative Suboptimality Gap:** We implement the certifiably-correct SE–Sync [8] to get the globally optimal objective value \( F^* \) of distributed PGO with the trivial loss kernel (Example 1), making it possible to compute the relative suboptimality gap \( (F - F^*)/F^* \) where \( F \) is the objective value for each iteration. The results are in Fig. 3.

**Riemannian Gradient Norm:** We also compute the Riemannian gradient norm of distributed PGO with the trivial, Huber, and Welsch loss kernels in Examples 1–3 for evaluation. Note that it is difficult to find globally optimal solutions to distributed PGO if Huber and Welsch loss kernels are used. The results are shown in Figs. 4–6.

In Figs. 3–6, it can be seen that MM–PGO, AMM–PGO*, AMM–PGO#, and AMM–PGO have a faster convergence if the number of robots (nodes) decreases. This is expected since \( G(X|X^{(k)}) \) and \( H(X|X^{(k)}) \) in (33) and (37) result in tighter approximations for distributed PGO with fewer robots (nodes). In addition, Figs. 4–6 suggest that the convergence rate of MM–PGO, AMM–PGO*, AMM–PGO#, and AMM–PGO also relies on the type of loss kernels. Nevertheless, AMM–PGO*, AMM–PGO#, and AMM–PGO accelerated by Nesterov’s method outperform unaccelerated MM–PGO by a large margin for any number of robots and any type of loss kernels, which means that Nesterov’s method improves the convergence of distributed PGO. In particular, Figs. 3(a)–6(a) indicate that AMM–PGO# with 50 robots still converges faster than MM–PGO with 5 robots despite that the later has a much smaller number of robots. Therefore, we conclude that Nesterov’s method accelerates the convergence of distributed PGO.

We emphasize the convergence comparisons of Nesterov’s accelerated AMM–PGO*, AMM–PGO#, and AMM–PGO that merely differ from each other by the adaptive restart schemes—AMM–PGO* has an additional master node to aggregate information from all the robots (nodes), whereas AMM–PGO# and AMM–PGO are restricted to one internode communication round per iteration among neighboring robots (nodes). Notwithstanding limited local communication, as is shown in Figs. 3, 5, and 6, AMM–PGO# has a convergence rate comparable to that of AMM–PGO* using a master node while being significantly faster than AMM–PGO. In particular, AMM–PGO# reduces adaptive restarts by 80%–95% compared to AMM–PGO on the Cube datasets and, thus, is expected to make better use of
Fig. 3. Relative suboptimality gaps of MM–PGO, AMM–PGO*, AMM–PGO#, and AMM–PGO [32] for distributed PGO with the trivial loss kernel on 5, 10, and 50 robots. The results are averaged over 20 Monte Carlo runs. (a) AMM–PGO# versus MM–PGO. (b) 5 robots. (c) 10 robots. (d) 50 robots.

Fig. 4. Riemannian gradient norms of MM–PGO, AMM–PGO*, AMM–PGO#, and AMM–PGO [32] for distributed PGO with the trivial loss kernel on 5, 10, and 50 robots. The results are averaged over 20 Monte Carlo runs. (a) AMM–PGO# versus MM–PGO. (b) 5 robots. (c) 10 robots. (d) 50 robots.

Fig. 5. Riemannian gradient norms of MM–PGO, AMM–PGO*, AMM–PGO#, and AMM–PGO [32] for distributed PGO with the Huber loss kernel on 5, 10, and 50 robots. The results are averaged over 20 Monte Carlo runs. (a) AMM–PGO# versus MM–PGO. (b) 5 robots. (c) 10 robots. (d) 50 robots.

Fig. 6. Riemannian gradient norms of MM–PGO, AMM–PGO*, AMM–PGO#, and AMM–PGO [32] for distributed PGO with the Welsch loss kernel on 5, 10, and 50 robots. The results are averaged over 20 Monte Carlo runs. (a) AMM–PGO# versus MM–PGO. (b) 5 robots. (c) 10 robots. (d) 50 robots.
Nesterov’s acceleration. Since AMM–PGO and AMM–PGO differ in the adaptive restart schemes, we attribute the faster convergence of AMM–PGO to its redesigned adaptive restart scheme. These results suggest that AMM–PGO is advantageous over other methods for very large-scale distributed PGO where computational and communicational efficiency are equally important.

B. Benchmark Datasets

In this section, we evaluate our MM methods (MM–PGO, AMM–PGO, and AMM–PGO) for distributed PGO on a number of 2-D and 3-D SLAM benchmark datasets [8] (see [43, Appendix L]). We use the trivial loss kernel and there are no outliers such that the globally optimal solution can be exactly computed with SE–Sync [8]. For each dataset, we also make comparisons against SE–Sync [8], distributed Gauss-Seidel (DGS) [36] and the Riemannian block coordinate descent (RBCD) [37] method, all of which are the state-of-the-art algorithms for centralized and distributed PGO. The SE–Sync and DGS methods use the recommended settings in [8] and [36]. We implement two Nesterov’s accelerated variants of RBCD [37], i.e., one with greedy selection rule and adaptive restart (RBCD++) and the other with uniform selection rule and fixed restart (RBCD+). As mentioned before, AMM–PGO and AMM–PGO can take at most one iteration when updating $X^{\alpha(k+1)}$ using (48) and (56), which is similar to RBCD++ and RBCD++. An overview of the aforementioned methods is given in Table I.

Number of Iterations: First, we examine the convergence of MM–PGO, AMM–PGO, AMM–PGO, DGS [36], RBCD++ [37], and RBCD++ [37] w.r.t. the number of iterations. The distributed PGO has 10 robots and all the methods are initialized with distributed Nesterov’s accelerated chorial initialization [32].

The objective values of each method with 100, 250, and 1000 iterations are reported in Table II and the reconstruction results using AMM–PGO are shown in Figs. 7 and 8. For almost all the benchmark datasets, AMM–PGO and AMM–PGO outperform the other methods (MM–PGO, DGS, RBCD++, and RBCD++). While RBCD++ and RBCD++ have similar performances in four relatively easy datasets—CSAIL, sphere, torus, and grid—AMM–PGO and AMM–PGO achieve much better results in the other more challenging datasets in particular if there are no more than 250 iterations. As discussed later, AMM–PGO and AMM–PGO have faster convergence to more accurate estimates without any extra computation and communication in contrast to RBCD++ and RBCD++. Last but not least, Table II demonstrates that the accelerated AMM–PGO and AMM–PGO converge significantly faster than the unaccelerated MM–PGO, which further validates the usefulness of Nesterov’s method.

We also compute the performance profiles [52] based on the number of iterations. Given a tolerance $\Delta \in (0, 1]$, the objective value threshold $F_{\Delta}(p)$ of a PGO problem $p$ is

$$F_{\Delta}(p) = F^* + \Delta \cdot \left( F^{(0)} - F^* \right)$$

where $F^{(0)}$ and $F^*$ are the initial and globally optimal objective values, respectively. Let $I_\Delta(p)$ denote the minimum number of iterations that a PGO method takes to reduce the objective value to $F_{\Delta}(p)$, i.e.,

$$I_\Delta(p) = \min_{k} \left\{ k \geq 0 | F^{(k)} \leq F_{\Delta}(p) \right\}$$

where $F^{(k)}$ is the objective value at iteration $k$. Then, for a problem set $P$, the performance profiles of a PGO method is the percentage of problems solved w.r.t. the number of iterations $k$

$$\text{percentage of problems solved } \Delta \left| \frac{|p \in P | I_\Delta(p) \leq k|}{|P|} \right.$$  

The performance profiles based on the number of iterations over a variety of 2-D and 3-D SLAM benchmark datasets (see [43, Appendix L]) are shown in Fig. 9. The tolerances evaluated are $\Delta = 1 \times 10^{-2}$, $5 \times 10^{-3}$, $1 \times 10^{-3}$, and $1 \times 10^{-4}$. We report the performance of MM–PGO, AMM–PGO, AMM–PGO, DGS [36], RBCD++ [37], and RBCD++ [37] for distributed PGO with 10 robots (nodes). As expected, AMM–PGO and AMM–PGO dominate the other methods (MM–PGO, DGS, RBCD++, and RBCD++) in terms of the convergence for all the tolerances $\Delta$, which means that both of them are better choices for distributed PGO.

In Table II and Fig. 9, we emphasize that AMM–PGO requiring no master node achieves comparable performance to that of AMM–PGO using a master node, and is a lot better than all the other methods with a master node (RBCD++) and without (MM–PGO, DGS and RBCD++). Even though RBCD++ and RBCD++ are similarly accelerated with Nesterov’s method, we remark that RBCD++ without a master node suffers a great performance drop compared to RBCD++, and more importantly, RBCD++ has no convergence guarantees to first-order critical points. These results reverify that AMM–PGO is more suitable for very large-scale distributed PGO with limited local communication.

---

**Table I**

Overview of the State-of-the-Art Algorithms for Distributed and Centralized PGO

| Method     | Distributed | Accelerated | Masterless | Converged |
|------------|-------------|-------------|------------|-----------|
| SE–Sync [8] | ×           | N/A         | N/A        | YES       |
| DGS [36]   | YES         | ×           | YES        | ×         |
| RBCD++ [37] | YES         | YES         | ×          | YES       |
| MM–PGO     | YES         | ×           | YES        | YES       |
| AMM–PGO*   | YES         | YES         | ×          | YES       |
| AMM–PGO#   | YES         | YES         | ×          | YES       |

Note that AMM–PGO* and RBCD++ require a master node for distributed PGO, and AMM–PGO# is the only accelerated method with provable convergence for distributed PGO without master node.

---

2 RBCD [37] solves the lifted problem, which usually results in slightly smaller objective values than the original problem.

3 In the experiments, we run RBCD++ [37] with fixed restart frequencies of 30, 50, and 100 iterations for each dataset and report the best results.
AMM–PGO results on the 2-D SLAM benchmark datasets where the different colors denote the odometries of different robots. The distributed PGO has 10 robots and is initialized with the distributed Nesterov’s accelerated chordal initialization [32]. The number of iterations is 1000. (a) ais2klinik. (b) city. (c) CSAIL. (d) M3500. (e) intel. (f) MITb.

AMM–PGO results on the 3-D SLAM benchmark datasets where the different colors denote the odometries of different robots. The distributed PGO has 10 robots and is initialized with the distributed Nesterov’s accelerated chordal initialization [32]. The number of iterations is 1000. (a) sphere. (b) torus. (c) grid. (d) garage. (e) cubicle. (f) rim.
TABLE II
RESULTS OF DISTRIBUTED PGO ON 2-D AND 3-D SLAM BENCHMARK DATASETS (SEE [43, APPENDIX B])

| Dataset | $F^{(0)}$ | $F^*$ | k | Methods w/ Master Node | Methods w/o Master Node |
|---------|-----------|--------|---|-------------------------|-------------------------|
| als2klinik | $3.8375 \times 10^2$ | $1.8850 \times 10^2$ | 100 | $2.0372 \times 10^2$ | $2.1079 \times 10^2$ | $2.1914 \times 10^2$ | $2.0371 \times 10^2$ | $8.4701 \times 10^2$ | $2.1715 \times 10^2$ |
| city | $7.0404 \times 10^2$ | $6.3862 \times 10^2$ | 100 | $6.4327 \times 10^2$ | $6.5138 \times 10^2$ | $6.5591 \times 10^2$ | $6.4327 \times 10^2$ | $7.7745 \times 10^2$ | $6.5962 \times 10^2$ |
| CSAI | $3.1719 \times 10^1$ | $3.1704 \times 10^1$ | 100 | $3.1704 \times 10^1$ | $3.1704 \times 10^1$ | $3.1704 \times 10^1$ | $3.1704 \times 10^1$ | $3.1704 \times 10^1$ | $3.1704 \times 10^1$ |
| M3S00 | $2.2311 \times 10^1$ | $1.9386 \times 10^2$ | 100 | $1.9446 \times 10^2$ | $1.9511 \times 10^2$ | $1.9560 \times 10^2$ | $1.9447 \times 10^2$ | $1.9557 \times 10^2$ | $1.9551 \times 10^2$ |
| intel | $5.3269 \times 10^1$ | $5.2348 \times 10^1$ | 100 | $5.2397 \times 10^1$ | $5.2496 \times 10^1$ | $5.2517 \times 10^1$ | $5.2397 \times 10^1$ | $5.2541 \times 10^1$ | $5.2526 \times 10^1$ |
| MITb | $8.8430 \times 10^2$ | $6.1154 \times 10^3$ | 100 | $6.1431 \times 10^1$ | $6.1518 \times 10^1$ | $6.1507 \times 10^1$ | $6.1330 \times 10^1$ | $5.9460 \times 10^1$ | $6.1997 \times 10^1$ |

3D SLAM Benchmark Datasets

| sphere | $1.9704 \times 10^3$ | $1.6870 \times 10^3$ | 100 | $1.6870 \times 10^3$ | $1.6870 \times 10^3$ | $1.6870 \times 10^3$ | $1.6870 \times 10^3$ | $1.6870 \times 10^3$ | $1.6870 \times 10^3$ |
| torus | $2.4654 \times 10^2$ | $2.4227 \times 10^4$ | 100 | $2.4227 \times 10^4$ | $2.4227 \times 10^4$ | $2.4227 \times 10^4$ | $2.4227 \times 10^4$ | $2.4227 \times 10^4$ | $2.4227 \times 10^4$ |
| grid | $2.8218 \times 10^2$ | $8.4319 \times 10^4$ | 100 | $8.4323 \times 10^4$ | $8.4320 \times 10^4$ | $8.3830 \times 10^4$ | $8.4399 \times 10^4$ | $8.4847 \times 10^4$ | $8.4920 \times 10^4$ |
| garage | $1.5470 \times 10^3$ | $1.2625 \times 10^6$ | 100 | $1.3105 \times 10^6$ | $1.3282 \times 10^6$ | $1.3396 \times 10^6$ | $1.3106 \times 10^6$ | $1.3170 \times 10^6$ | $1.3364 \times 10^6$ |
| cubic | $8.3514 \times 10^3$ | $7.1713 \times 10^5$ | 100 | $7.1812 \times 10^5$ | $7.2048 \times 10^5$ | $7.2300 \times 10^5$ | $7.1812 \times 10^5$ | $7.2185 \times 10^5$ | $7.2210 \times 10^5$ |
| rim | $8.1406 \times 10^4$ | $5.4609 \times 10^3$ | 100 | $5.5044 \times 10^3$ | $5.7184 \times 10^3$ | $5.8138 \times 10^3$ | $5.5044 \times 10^3$ | $5.8140 \times 10^3$ | $5.7810 \times 10^3$ |
| | | | 250 | $5.4648 \times 10^3$ | $5.5050 \times 10^3$ | $5.7197 \times 10^3$ | $5.4648 \times 10^3$ | $5.6184 \times 10^3$ | $5.7195 \times 10^3$ |
| | | | | $5.4609 \times 10^3$ | $5.4617 \times 10^3$ | $5.5509 \times 10^3$ | $5.4609 \times 10^3$ | $5.6258 \times 10^3$ | $5.5373 \times 10^3$ |

The distributed PGO has 10 robots and is initialized with distributed distributed chordal initialization [32]. We report the objective values of each method with 100, 250 and 1000 iterations. $F^{(0)}$ and $F^*$ are the objective value at iteration 0 and globally optimal objective value, respectively. The best results are colored in red and the second best in blue if no methods tie for the The distributed PGO has 10 robots and is initialized with distributed distributed chordal initialization [32]. We report the objective values of each method with 100, 250 and 1000 iterations. $F^{(0)}$ and $F^*$ are the objective value at iteration 0 and globally optimal objective value, respectively. The best results are colored in red and the second best in blue if no methods tie for the best

Note that all of MM–PGO, AMM–PGO*, AMM–PGO#, DGS [36], RBBC++ [37], and RBBC++# [37] have to exchange poses of internode measurements with the neighbors and, thus, need almost the same amount of communication per iteration. However, Fig. 9 indicates that AMM–PGO* and AMM–PGO# have much faster convergence in terms of the number of iterations, which also means less communication for the same level of accuracy. In addition, RBBC+++ and RBBC++# have to keep part of the nodes idle during optimization and rely on red–black coloring for block aggregation and random sampling for block selection, which induce additional computation and communication. In contrast, neither AMM–PGO* nor AMM–PGO# has any extra practical restrictions except Assumptions 1–4.

**Optimization Time:** We evaluate the optimization time of AMM–PGO and AMM–PGO# with different numbers of robots (nodes) against the centralized baseline SE–Sync [8]. To improve the time efficiency of our methods, $X^{(n)(k+1)}$ in (48) and (56) uses the same rotation as $X^{(n)(k+1/2)}$ in (47) and (55) and merely updates the translation. Due to the different numbers of robots (nodes), the centralized chordal initialization [40] is used for all the runs.
Similar to the number of iterations, we use the performance profiles to evaluate AMM–PGO* and AMM–PGO# in terms of the optimization time. Recall from (69) the objective value threshold $F_\Delta(p)$ where $p$ is the PGO problem and $\Delta \in [0, 1]$ is the tolerance. Since the average optimization time per node is directly related to the speedup, we measure the efficiency of a distributed PGO method with $N$ nodes by computing the average optimization time per node $T_\Delta(p, \tilde{N})$ that each node takes to reduce the objective value to $F_\Delta(p)$

$$T_\Delta(p, \tilde{N}) = \frac{T_\Delta(p)}{\tilde{N}}$$  \hspace{1cm} (70)$$

where $T_\Delta(p)$ denotes the total optimization time of all the $N$ nodes. We remark that the centralized optimization method has $\tilde{N} = 1$ node and $T_\Delta(p, \tilde{N}) = T_\Delta(p)$. Let $T_{\text{SE-Sync}}$ denote the optimization time that SE-Sync needs to find the globally optimal solution. The performance profiles assume a distributed PGO method solves problem $p$ for some $\mu \in [0, +\infty)$ if $T_\Delta(p, \tilde{N}) \leq \mu \cdot T_{\text{SE-Sync}}$. Note that $\mu$ is the scaled average optimization time per node and SE–Sync solves problem $p$ globally at $\mu = 1$.

As a result in [52], the performance profiles evaluate the speedup of distributed PGO methods for a given optimization problem set $\mathcal{P}$ using the percentage of problems solved w.r.t.

$$\% \text{probs. solved} = \frac{\sum_{p \in \mathcal{P}} \mathbb{1} \left\{ T_\Delta(p, \tilde{N}) \leq \mu \cdot T_{\text{SE-Sync}} \right\}}{|\mathcal{P}|} \times 100$$

where $\mathbb{1} \left\{ \cdot \right\}$ is the indicator function.

Our method, due to the optimization method taking distribution into account, can be parallelized while retaining guarantees on convergence and computation. A comparison using $T_{\text{SE-Sync}}$ assumes there is no value in parallelization, and, indeed, in that

---

**Fig. 9.** Performance profiles for MM–PGO, AMM–PGO*, AMM–PGO#, DGS [36], RBBCD++* [37], and RBBCD++# [37] on 2-D and 3-D SLAM Benchmark datasets (see [43, Appendix L]). The performance is based on the number of iterations $k$ and the evaluation tolerances are $\Delta = 1 \times 10^{-2}$, $5 \times 10^{-3}$, $1 \times 10^{-3}$, and $1 \times 10^{-4}$. The distributed PGO has 10 robots (nodes) and is initialized with distributed Nesterov’s accelerated chordal initialization [32]. Note that AMM–PGO* and RBBCD++* require a master node, whereas MM–PGO, AMM–PGO#, DGS, and RBBCD++# do not. (a) $\Delta = 1 \times 10^{-2}$, (b) $\Delta = 5 \times 10^{-3}$, (c) $\Delta = 1 \times 10^{-3}$, (d) $\Delta = 1 \times 10^{-4}$.

**Fig. 10.** Performance profiles for AMM–PGO*, AMM–PGO#, and SE–Sync [8] on 2-D and 3-D SLAM benchmark datasets (see [43, Appendix L]). The performance is based on the scaled average optimization time per node $\mu \in [0, +\infty)$ with tolerances $\Delta = 1 \times 10^{-2}$, $1 \times 10^{-3}$, $1 \times 10^{-4}$, and $1 \times 10^{-5}$. The distributed PGO has 10, 25, and 100 robots (nodes) and is initialized with the centralized chordal initialization [40]. Note that SE–Sync solves all the PGO problems globally at $\mu = 1$. (a) $\Delta = 1 \times 10^{-2}$, (b) $\Delta = 1 \times 10^{-3}$, (c) $\Delta = 1 \times 10^{-4}$, (d) $\Delta = 1 \times 10^{-5}$.

**Fig. 11.** ATEs of distributed PGO using AMM–PGO# with the trivial, Huber, and Welch loss kernels on the 2-D intel and 3-D garage datasets. The outlier ratios of inter-node loop closures are 0 – 0.9. The ATEs are computed against the outlier-free results of SE–Sync [8] and are averaged over 10 Monte Carlo runs. PCM [53] is used to initially reject spurious loop closures. (a) intel, (b) garage.
setting, SE–Sync would be competitive with our method. But parallelization is valuable, and the $T_{\Delta}(p, N) = T_{\Delta}(p) / N$ metric in (70) captures that value and shows that when we distribute the optimization across agents we get performance that is both superior in accuracy and faster.

Fig. 10 shows the performance profiles based on the scaled average optimization time per node. The tolerances evaluated are $\Delta = 1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, \text{and} \ 1 \times 10^{-5}$. We report the performance of AMM–PGO* and AMM–PGO# with 10, 25, and 100 robots (nodes). For reference, we also evaluate the performance profile of the centralized PGO baseline SE–Sync [8]. As the results demonstrate, AMM–PGO* and AMM–PGO# are significantly faster than SE–Sync [8] in most cases for modest accuracies of $\Delta = 1 \times 10^{-2}$ and $\Delta = 1 \times 10^{-3}$, for which the only challenging case is the CSAIL dataset, whose chordal initialization is already very close to the globally optimal solution. In spite of the performance decline for smaller tolerances of $\Delta = 1 \times 10^{-4}$ and $\Delta = 1 \times 10^{-5}$, AMM–PGO* and AMM–PGO# with 100 robots (nodes) still achieve a 2.5–20x speedup of optimization time over SE–Sync for more than 70% of the benchmark datasets, not to mention that the average optimization time per node of AMM–PGO* and AMM–PGO# decreases with the number of robots (nodes). Note that the communication overhead is not considered in the experiments. Nevertheless, Fig. 10 indicates that AMM–PGO* and AMM–PGO# are promising as fast parallel backends for very large-scale PGO and real-time multirobot SLAM.

In summary, AMM–PGO* and AMM–PGO# achieve the state-of-the-art performance for distributed PGO and enjoy a significant multidone speedup compared to the centralized baseline [8] for modestly but sufficiently accurate estimates.

C. Robust Distributed PGO

In this section, we evaluate the robustness of AMM–PGO# against the outlier internode loop closures. Similar to the work in [24] and [27], we first use the distributed pairwise consistent measurement set maximization algorithm (PCM) [53] to reject spurious internode loop closures and then solve the resulting distributed PGO using AMM–PGO# with the trivial, Huber, and Welsch loss kernels in Examples 1–3.

We implement AMM–PGO# on the 2-D intel and 3-D garage datasets (see [43, Appendix L]) with 10 robots (nodes). For each dataset, we add false internode loop closures with uniformly random rotation and translation errors in the range of $[0, \pi]$ rad and $[0, 5]$ m, respectively. In addition, after the initial outlier rejection using the PCM algorithm [53], we initialize AMM–PGO# with distributed Nesterov’s accelerated chordal initialization [32] for all the loss kernels.

The absolute trajectory errors (ATEs) of AMM–PGO# w.r.t. different outlier ratios of internode loop closures are in Fig. 11. The ATEs are computed against the outlier-free results of SE–Sync [8] and averaged over 10 Monte Carlo runs.

In Fig. 11(a), PCM [53] rejects most of the outlier inter-node loop closure for the intel dataset and AMM–PGO# solves the distributed PGO problems regardless of the loss kernel types and outlier ratios. Note that AMM–PGO# with the Welsch loss kernel has larger ATEs (avg. 0.057 m) against SE–Sync [8] than those with the trivial and Huber loss kernels (avg. 0.003 m), and we argue that this is related to the loss kernel types. The ATEs are evaluated based on SE–Sync using the trivial loss kernel, which is in fact identical/similar to distributed PGO with the trivial and Huber loss kernels but different from that with the Welsch loss kernel. Thus, the estimates from the trivial and Huber loss kernels are expected to be more close to those of SE–Sync, which result in smaller ATEs compared to the Welsch loss kernel if no outliers.

For the more challenging garage dataset, as is shown in Fig. 11(b), PCM fails for outlier ratios over 0.4, and further, distributed PGO with the trivial and Huber loss kernels results in ATEs as large as 65 m. In contrast, distributed PGO with the Welsch loss kernel still successfully estimates the poses with an average ATE of 2.5 m despite the existence of outliers—note that the garage dataset has a trajectory over 7 km. For the garage dataset, a qualitative comparison of distributed PGO with different loss kernels is also presented in Fig. 12, where the Welsch loss kernel still has the best performance. The results are not surprising since the Welsch loss kernel is known to be more robust against outliers than the other two loss kernels [46].

The results above indicate that our MM methods can be applied to distributed PGO in the presence of outlier internode loop closures when combined with robust loss kernels, such as Welsch, and other outlier rejection techniques, such as PCM [53]. In addition, we emphasize again that our MM methods have provable convergence to first-order critical points for a broad
class of robust loss kernels, whereas the convergence guarantees of existing distributed PGO methods [36, 37, 38, 39] are restricted to the trivial loss kernel.

XI. CONCLUSION AND FUTURE WORK

We presented MM methods for distributed PGO that has important applications in multirobot SLAM. Our MM methods had provable convergence for a broad class of robust loss kernels in robotics and computer vision. Furthermore, we elaborated on the use of Nesterov’s method and adaptive restart for acceleration and developed accelerated MM methods AMM–PGO and AMM–PGO* without sacrificing convergence guarantees. In particular, we designed a novel adaptive restart scheme making AMM–PGO* without a master node comparable to AMM–PGO* using a master node for information aggregation. The extensive experiments on numerous 2-D and 3-D SLAM datasets indicated that our MM methods outperformed existing state-of-the-art methods and robustly handled distributed PGO with outlier internode loop closures.

Our MM methods for distributed PGO can be improved as follows. A more tractable and robust initialization technique is definitely beneficial to the accuracy and efficiency of distributed PGO. Even though our MM methods have reliable performances against outliers, a more complete theoretical analysis for robust distributed PGO is still necessary. We might also extend our MM methods for differentiable distributed PGO [54]. In addition, our MM methods can be implemented as local solvers for distributed certifiably correct PGO [37] to handle poor or random initialization. Since all the nodes are now assumed to be synchronized, it is necessary and useful to extend our MM methods for asynchronous distributed PGO. Finally, real multirobot tests might make the results of our MM methods more convincing where not only the optimization time but also the communication overhead can be validated.

ACKNOWLEDGMENT

The authors thank Yulun Tian for sharing the code of the Riemannian block coordinate descent method for distributed PGO.

REFERENCES

[1] C. Cadena et al., “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” IEEE Trans. Robot., vol. 32, no. 6, pp. 1309–1332, Dec. 2016.
[2] D. M. Rosen, K. J. Doherty, A. T. Espinoza, and J. J. Leonard, “Advances in inference and representation for simultaneous localization and mapping,” Annu. Rev. Control, Robot., Auton. Syst., vol. 4, pp. 215–242, 2021.
[3] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Cambridge, MA, USA: MIT Press, 2005.
[4] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2012, pp. 3354–3361.
[5] A. Singer, “Angular synchronization by eigenvectors and semidefinite programming,” Appl. Comput. Harmon. Anal., vol. 30, no. 1, pp. 20–36, 2011.
[6] A. Singer and Y. Shkolnisky, “Three-dimensional structure determination from common lines in Cryo-EM by eigenvectors and semidefinite programming,” SIAM J. Imag. Sci., vol. 4, no. 2, pp. 543–572, 2011.
[7] D. M. Rosen, M. Kaess, and J. J. Leonard, “RISE: An incremental trust-region method for robust online sparse least-squares estimation,” IEEE Trans. Robot., vol. 30, no. 5, pp. 1091–1108, Oct. 2014.
[8] 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,” Int. J. Robot. Res., vol. 38, no. 2/3, pp. 95–125, 2019.
[9] D. M. Rosen, “Scalable low-rank semidefinite programming for certifiably correct machine perception,” in Proc. Int. Workshop Algorithmic Found. Robot., 2020, vol. 3, pp. 551–566.
[10] F. Dellaert, “Factor graphs and SAM: A hands-on introduction,” Georg. Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-RIM-CP&R-2012-002, 2012.
[11] T. Fan, H. Wang, M. Rubenstein, and T. Murphey, “Efficient and guaranteed planar pose graph optimization using the complex number representation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2019, pp. 1904–1911.
[12] T. Fan, H. Wang, M. Rubenstein, and T. Murphey, “CPL-SLAM: Efficient and certifiably correct planar graph-based SLAM using the complex number representation,” IEEE Trans. Robot., vol. 36, no. 6, pp. 1719–1737, Dec. 2020.
[13] L. Carlone, G. C. Calafiore, C. Tommillo, and F. Dellaert, “Planar pose graph optimization: Duality, optimal solutions, and verification,” IEEE Trans. Robot., vol. 32, no. 3, pp. 545–565, Jun. 2016.
[14] L. Carlone, D. M. Rosen, G. Calafiore, J. J. Leonard, and F. Dellaert, “Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2015, pp. 125–132.
[15] G. Grisetti, C. Stachniss, and W. Burgard, “Nonlinear constraint network optimization for efficient map learning,” IEEE Trans. Intell. Transp. Syst., vol. 10, no. 3, pp. 428–439, Sep. 2009.
[16] J. Biales and J. Gonzalez-Izunza, “Cartan-Sync: Fast and global SE(d)-synchronization,” IEEE Robot. Autom. Lett., vol. 2, no. 4, pp. 2127–2134, Oct. 2017.
[17] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell, “Coordinated multi-robot planning while preserving individual privacy,” in Proc. Int. Conf. Robot. Automat., 2019, pp. 2188–2194.
[18] Y. Zhang and D. A. Shell, “Complete characterization of a class of privacy-preserving tracking problems,” Int. J. Robot. Res., vol. 38, no. 2/3, pp. 299–315, 2019.
[19] A. Cunningham, M. Paluri, and F. Dellaert, “DDF-SAM: Fully distributed SLAM using constrained factor graphs,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2010, pp. 3025–3030.
[20] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues, “Multi-agent localization from noisy relative pose measurements,” in Proc. IEEE Int. Conf. Robot. Automat., 2011, pp. 364–369.
[21] A. Cunningham, V. Indelman, and F. Dellaert, “DDF-SAM 2.0: Consistent distributed smoothing and mapping,” in Proc. IEEE Int. Conf. Robot. Automat., 2013, pp. 5220–5227.
[22] S. Sacedi, M. Trentini, M. Seto, and H. Li, “Multiple-robot simultaneous localization and mapping: A review,” J. Field Robot., vol. 33, no. 1, pp. 3–46, 2016.
[23] J. Dong, E. Nelson, V. Indelman, M. Michael, and F. Dellaert, “Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach,” in Proc. IEEE Int. Conf. Robot. Automat., 2015, pp. 5807–5814.
[24] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, “DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 1656–1663, Apr. 2020.
[25] T. Cieslewski, S. Choudhary, and D. Scaramuzza, “Data-efficient decentralized visual SLAM,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 2466–2473.
[26] V. Tchuev and V. Indelman, “Distributed consistent multi-robot semantic localization and mapping,” IEEE Robot. Autom. Lett., vol. 5, no. 3, pp. 4649–4656, Jul. 2020.
[27] 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 Proc. IEEE Int. Conf. Robot. Automat., 2021, pp. 11210–11218.
[28] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, “Kimera-Multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems,” IEEE Trans. Robot., vol. 38, no. 4, pp. 2022–2038, Aug. 2022.
[29] D. R. Hunter and K. Lange, “A tutorial on MM algorithms,” Amer. Statistician, vol. 58, no. 1, pp. 30–37, 2004.
[30] Y. Sun, P. Babu, and D. P. Palomar, “Majorization-minimization algorithms in signal processing, communications, and machine learning,” IEEE Trans. Signal Process., vol. 65, no. 3, pp. 794–816, Feb. 2017.

[31] T. Fan and T. Murphey, “Generalized proximal methods for pose graph optimization,” in Proc. Int. Symp. Robot. Res., 2019, pp. 393–409.

[32] T. Fan and T. Murphey, “Majorization minimization methods for distributed pose graph optimization with convergence guarantees,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots. Syst., 2020, pp. 5058–5065.

[33] Y. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence O (1/k^2),” Dokl. Akad. Nauk. SSSR, vol. 269, pp. 543–547, 1983.

[34] Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, vol. 87. Berlin, Germany: Springer, 2013.

[35] B. O’donoghue and E. Candès, “Adaptive restart for accelerated gradient schemes,” Found. Comput. Math., vol. 15, no. 3, pp. 715–732, 2015.

[36] S. Choudhary et al., “Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models,” Int. J. Robot. Res., vol. 36, no. 12, pp. 1286–1311, 2017.

[37] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, “Distributed certifiably correct pose-graph optimization,” IEEE Trans. Robot., vol. 37, no. 6, pp. 2137–2156, Dec. 2021.

[38] R. Tron and R. Vidal, “Distributed 3-D localization of camera sensor networks from 2-D image measurements,” IEEE Trans. Autom. Control, vol. 59, no. 12, pp. 3325–3340, Dec. 2014.

[39] E. Cristofalo, E. Montijano, and M. Schwager, “GeoD: Consensus-based geodesic distributed pose graph optimization,” 2020, arXiv:2010.00156.

[40] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, “Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization,” in Proc. IEEE Int. Conf. Robot. Automat., 2015, pp. 4597–4604.

[41] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, “Asynchronous and parallel distributed pose graph optimization,” IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 5819–5826, Oct. 2020.

[42] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, “Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 1127–1134, Apr. 2020.

[43] T. Fan and T. Murphey, “Majorization minimization methods for distributed pose graph optimization,” 2021, arxiv:2108.00083.

[44] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robust map optimization using dynamic covariance scaling,” in Proc. IEEE Int. Conf. Robot. Automat., 2013, pp. 62–69.

[45] L. Carlone and G. C. Calafiore, “Convex relaxations for pose graph optimization with outliers,” IEEE Robot. Autom. Lett., vol. 3, no. 2, pp. 1160–1167, Apr. 2018.

[46] J. T. Barron, “A general and adaptive robust loss function,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4326–4334.

[47] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 13, no. 4, pp. 376–380, Apr. 1991.

[48] S. Ghadimi and G. Lan, “Accelerated gradient methods for nonconvex nonlinear and stochastic programming,” Math. Program., vol. 156, no. 1/2, pp. 59–99, 2016.

[49] C. Jin, P. Netrapalli, and M. I. Jordan, “Accelerated gradient descent escapes saddle points faster than gradient descent,” in Proc. Conf. Learn. Theory, 2018, pp. 1042–1085.

[50] H. Li and Z. Lin, “Accelerated proximal gradient methods for nonconvex programming,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 379–387.

[51] H. Zhang and W. W. Hager, “A nonmonotone line search technique and its application to unconstrained optimization,” SIAM J. Optim., vol. 14, no. 4, pp. 1043–1056, 2004.

[52] E. D. Dolan and J. J. Moré, “Benchmarking optimization software with performance profiles,” Math. Program., vol. 91, no. 2, pp. 201–213, 2002.

[53] J. G. Mangelson, D. Dottinie, R. M. Eustice, and R. Vasudevan, “Pairwise consistent measurement set maximization for robust multi-robot map merging,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 2916–2923.

[54] L. Pineda et al., “Theseus: A library for differentiable nonlinear optimization,” in Proc. Neural Inf. Process. Syst., 2022, pp. 3801–3818.

Taoshia Fan received the B.E. degree in automotive engineering from Tongji University, Shanghai, China, in 2013, the M.S. degree in mechanical engineering and mathematics from Johns Hopkins University, Baltimore, MD, USA, in 2015, and the Ph.D. degree in mechanical engineering from Northwestern University, Evanston, IL, USA, in 2022. He is currently a Research Engineer with Meta AI, Pittsburgh, PA, USA. His research interests include the intersection of robotic control, simulation, and estimation.

Todd D. Murphey (Senior Member, IEEE) received the B.S. degree in mathematics from the University of Arizona, Tucson, AZ, USA, in 1997, and the Ph.D. degree in control and dynamical systems from the California Institute of Technology, Pasadena, CA, USA, in 2002.

He is currently a Professor of Mechanical Engineering with Northwestern University, Evanston, IL, USA. His laboratory is part of the Center for Robotics and Biosystems. His research interests include robotics, control, machine learning in physical systems, and computational neuroscience.