Distributed Certifiably Correct Pose-Graph Optimization

Yulun Tian, Kasra Khosoussi, David M. Rosen, and Jonathan P. How

***Abstract*—This article presents the first *certifiably correct* algorithm for *distributed* pose-graph optimization (PGO), the backbone of modern collaborative simultaneous localization and mapping (CSLAM) and camera network localization (CNL) systems. Our method is based upon a sparse semidefinite relaxation that we prove provides globally optimal PGO solutions under moderate measurement noise (matching the guarantees enjoyed by the state-of-the-art centralized methods), but is amenable to distributed optimization using the low-rank Riemannian Staircase framework. To implement the Riemannian Staircase in the distributed setting, we develop *Riemannian block coordinate descent* (RBCD), a novel method for (locally) minimizing a function over a product of Riemannian manifolds. We also propose the first distributed solution verification and saddle escape methods to certify the global optimality of critical points recovered via RBCD, and to descend from suboptimal critical points (if necessary). All components of our approach are inherently decentralized: they require only local communication, provide privacy protection, and are easily parallelizable. Extensive evaluations on synthetic and real-world datasets demonstrate that the proposed method correctly recovers globally optimal solutions under moderate noise, and outperforms alternative distributed techniques in terms of solution precision and convergence speed.**

***Index Terms*—Multi-robot systems, optimization, simultaneous localization and mapping.**

I. Introduction

COLLABORATIVE multirobot missions require *consistent collective* spatial perception across the entire team. In unknown GPS-denied environments, this is achieved by *collaborative* simultaneous localization and mapping (CSLAM), in which a team of agents *jointly* constructs a *common* model of an environment via exploration. At the heart of CSLAM, robots must solve a *pose-graph optimization* (PGO) problem (also known as *pose synchronization*) to estimate their trajectories based on noisy relative *interrobot* and *intrarobot* measurements.

While several prior approaches to CSLAM have appeared in the literature, to date no method has been proposed that is capable of *guaranteeing* the recovery of an optimal solution in the distributed setting. One general line of prior work [2]–[5] proposes to solve CSLAM in a *centralized* manner. While this approach is straightforward (as it enables the use of off-the-shelf methods for PGO), it imposes several practically restrictive requirements, including: a central node that is capable of solving the *entire team's* SLAM problem, a sufficiently reliable communication channel that connects the central node to the team, and sufficient resources (particularly energy and bandwidth) to regularly relay the team's (raw or processed) observations to the central node. Moreover, these schemes *by construction* are unable to protect the *spatial privacy* of individual robots, and lack robustness due to having a single point of failure. An alternative line of work proposes fully distributed algorithms [6]–[9]; however, at present these methods are all based upon applying distributed *local* optimization methods to the *nonconvex* PGO problem [6], [10], which renders them vulnerable to convergence to significantly suboptimal solutions [11].

In parallel, recent work over the last several years has developed a novel class of *certifiably correct* estimation methods that *are* capable of efficiently recovering *provably globally optimal* solutions of generally intractable estimation problems within a restricted (but still practically relevant) operational regime [12]. In particular, Rosen *et al.* [13] developed SE-Sync, a certifiably correct algorithm for PGO. SE-Sync is based upon a (convex) semidefinite relaxation whose minimizer is *guaranteed* to provide a *globally optimal* PGO solution whenever the noise on the available measurements falls below a certain critical threshold; moreover, in the (typical) case that this occurs, it is possible to *computationally verify* this fact *a posteriori*, thereby *certifying* the correctness (optimality) of the recovered estimate. However, SE-Sync is not directly amenable to a *distributed* implementation because its semidefinite relaxation generically has *dense* data matrices, and it solves this relaxation using a second-order optimization scheme, both of which would require an impractical degree of communication among distributed agents.

In this article, we advance the state of the art in CSLAM by proposing the first PGO algorithm that is both *fully distributed* and *certifiably correct*. Our method leverages the same semidefinite relaxation strategy that underpins current state-of-the-art (centralized) certifiably correct PGO algorithms [13], but

Manuscript received September 1, 2020; revised February 20, 2021; accepted March 9, 2021. Date of publication May 7, 2021; date of current version December 6, 2021. This work was supported in part by the NASA Convergent Aeronautics Solutions project Design Environment for Novel Vertical Lift Vehicles (DELIVER), in part by ONR under BRC Award N000141712072, and in part by ARL DCIST under Cooperative Agreement W911NF-17-2-0181. This article was recommended for publication by Associate Editor S.-J Chung and Editor F. Chaumette upon evaluation of the reviewers' comments. *(Corresponding author: Yulun Tian.)*

The authors are with the Laboratory for Information, and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: yulun@mit.edu; kasra.mail@gmail.com; dmrosen@mit.edu; jhow@mit.edu, Fellow).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2021.3072346.

Digital Object Identifier 10.1109/TRO.2021.3072346

employs novel distributed optimization and solution verification techniques that enable these relaxations to be solved efficiently in the distributed setting. Specifically, we make the following contributions.

1) We prove that a sparse semidefinite relaxation of PGO employed by Briales and Gonzalez-Jimenez [14] enjoys the same exactness guarantees as the one used in SE-Sync [13]: namely, that its minimizers are *low-rank* and provide *exact* solutions of the original PGO problem under moderate measurement noise.
2) We describe an efficient low-rank optimization scheme to solve this semidefinite relaxation in the distributed setting. Specifically, we employ a *distributed* Riemannian Staircase approach [15] and propose *Riemannian block coordinate descent* (RBCD), a novel method for minimizing a function over a product of Riemannian manifolds, to solve the resulting low-rank subproblems in the distributed setting. We prove that RBCD converges to first-order critical points with a *global* sublinear rate under standard (mild) conditions, and that these are in particular *always* satisfied for the low-rank PGO subproblems. We also propose and analyze Nestorov-accelerated variants of RBCD that significantly improve its convergence speed in practice.
3) We propose the first *distributed solution verification* and *saddle escape* methods to certify the optimality of low-rank critical points recovered via RBCD, and to descend from suboptimal critical points (if necessary).
4) Finally, we describe simple distributed procedures for initializing the distributed Riemannian Staircase optimization, and for *rounding* the resulting low-rank factor to extract a final PGO estimate.

Each of these algorithmic components has the same communication, computation, and privacy properties enjoyed by current distributed CSLAM methods [6], [8], [9], including the following.

1) *Communication and computational efficiency:* Robots need only to communicate with their neighbors in the pose graph. To this end, the minimum requirement is robots form a *connected* network, so that information can flow between any pair of robots (possibly relayed by intermediate robots). The message size in each round of communication is only $\mathcal{O}(m_{\text{inter}})$, where m_{inter} is the number of interrobot loop closures. Moreover, local updates in RBCD can be performed efficiently and in *parallel*, and the solution is produced in an anytime fashion.
2) *Spatial privacy protection:* Robots are not required to reveal *any* information about their own observations or their *private* poses (those poses that are not *directly* observed by other robots).

Our overall algorithm, *Distributed Certifiably Correct Pose-Graph Optimization* (DC2-PGO), thus, preserves the desirable computational properties of existing state-of-the-art CSLAM methods while enabling the recovery of *provably globally optimal* solutions in the distributed setting.

The rest of this article is organized as follows. In the remainder of this section, we introduce necessary notations and mathematical preliminaries. In Section II, we review the state-of-the-art centralized and distributed PGO solvers, as well as recent advances in block-coordinate optimization methods. In Section III, we formally define the distributed PGO problem and present its sparse SDP relaxation. We present a distributed procedure to solve this SDP via the Riemannian Staircase framework. On the theoretical front, we establish formal *exactness* guarantees for the SDP relaxation. In Section IV, we present our distributed local search method to solve the rank-restricted SDPs using block-coordinate descent (BCD). In Section V, we present a distributed solution verification procedure that checks the global optimality of our local solutions, and enables us to *escape* from suboptimal critical points, if necessary. We discuss distributed initialization and rounding in Section VI. Finally, we conclude with extensive experimental evaluations in Section VII. Proofs and additional details, discussions, and experiments are provided in our extended technical report [1].

Notations and Preliminaries

General Notations: Unless stated otherwise, lowercase and uppercase letters are generally used for vectors and matrices, respectively. We define $[n] \triangleq \{1, 2, \ldots, n\}$.

Linear Algebra: $\mathbb{S}^d$ and $\mathbb{S}^d_+$ denote the set of $d \times d$ symmetric and symmetric positive semidefinite matrices, respectively. $0_{d\times d}, I_{d\times d} \in \mathbb{R}^{d\times d}$ are the zero and identity matrix, and $0_d, 1_d \in \mathbb{R}^d$ represent the vectors of all zeros and all ones, respectively. To ease the burden of notations, we also drop the subscript d when the dimension is clear from the context. For a matrix A, we use $A_{(i,j)}$ to index its (i, j)th entry. $A^\dagger$ denotes the Moore–Penrose inverse of A. Given a $(d \times d)$-block-structured matrix $Z \in \mathbb{R}^{dn\times dn}$, $Z_{[i,j]} \in \mathbb{R}^{d\times d}$ refers to its (i, j)th block. Following [13], we define $\mathrm{BlockDiag}(M)$ as the linear operator that extracts the diagonal blocks of M and zeros out all remaining blocks, and $\mathrm{SymBlockDiag}_d(M)$ as its symmetric version; see [13, Equations (4)-(5)]. $\mathrm{Proj}_{\mathcal{S}}$ denotes the orthogonal projection operator onto a given set $\mathcal{S}$ with respect to the Frobenius norm.

We define several linear operators that will be useful in the rest of this article. Given a list of square matrices $A_1, \ldots, A_n$ (possibly with different dimensions), Diag assembles them into a block-diagonal matrix. For an arbitrary square matrix A, Sym returns its symmetric part $\mathrm{Sym}(A) \triangleq (A + A^\top)/2$.

For a $[(d + 1) \times (d + 1)]$-block-structured matrix $Z \in \mathbb{R}^{(d+1)n\times(d+1)n}$, we define the following linear operator:

$$\mathrm{SymBlockDiag}^+_d(Z)_{[i,j]} \triangleq \begin{cases} \begin{bmatrix} \mathrm{Sym}(Z_{[i,i](1:d,1:d)}) & 0_d \\ 0_d^\top & 0 \end{bmatrix}, & \text{if } i = j, \\ 0_{(d+1)\times(d+1)}, & \text{o.w.} \end{cases} \tag{1}$$

Differential Geometry of Riemannian Manifolds: We refer the reader to [16] and [17] for outstanding introductions to Riemannian optimization on matrix submanifolds. In general, we use $\mathcal{M}$ to denote a smooth matrix submanifold of a real Euclidean space. $T_x\mathcal{M}$ (or T_x for brevity) denotes the tangent space at $x \in \mathcal{M}$. The tangent space is endowed with

the standard Riemannian metric induced from the ambient (Euclidean) space, i.e., $\langle \eta_1, \eta_2 \rangle \triangleq tr(\eta_1^\top \eta_2)$, and the induced norm is $\|\eta\| \triangleq \sqrt{\langle \eta, \eta \rangle}$. Retr denotes a retraction operator, with $\mathrm{Retr}_x : T_x\mathcal{M} \to \mathcal{M}$ being its restriction to $T_x\mathcal{M}$. For a function $f : \mathcal{M} \to \mathbb{R}$, we use $\nabla f(x)$ and $\mathrm{grad}\, f(x)$ to denote the Euclidean and Riemannian gradients of f at $x \in \mathcal{M}$. For matrix submanifolds, the Riemannian gradient is obtained as the orthogonal projection of the Euclidean gradient onto the associated tangent space:

$$\mathrm{grad}\, f(x) = \mathrm{Proj}_{T_x}(\nabla f(x)). \tag{2}$$

We call $x^\star \in \mathcal{M}$ a *first-order critical point* if $\mathrm{grad}\, f(x^\star) = 0$. For the second-order geometry, we are primarily concerned with the Riemannian Hessian of a function $f : \mathcal{M} \to \mathbb{R}$. For matrix submanifolds, the Riemannian Hessian is a linear mapping on the tangent space, which captures the directional derivative of the Riemannian gradient:

$$\begin{aligned} \mathrm{Hess}\, f(x) :\ & T_x\mathcal{M} \to T_x\mathcal{M}, \\ & \eta \mapsto \mathrm{Proj}_{T_x}(\mathrm{D}\,\mathrm{grad}\, f(x)[\eta]). \end{aligned} \tag{3}$$

Above, the operator D denotes the standard directional derivative in the Euclidean space; see [17, Ch. 5].

Matrix submanifolds used in this work: In SLAM, standard matrix submanifolds that frequently appear include the special orthogonal group $\mathrm{SO}(d) \triangleq \{R \in \mathbb{R}^{d\times d} | R^\top R = I_d, \det(R) = 1\}$ and the special Euclidean group $\mathrm{SE}(d) \triangleq \{(R, t) | R \in \mathrm{SO}(d), t \in \mathbb{R}^d\}$. We also make use of the Stiefel manifold $\mathrm{St}(d, r) \triangleq \{Y \in \mathbb{R}^{r\times d} \mid Y^\top Y = I_d\}$. The geometry of these manifolds can be found in standard textbooks (e.g., [17]). Given a matrix $A \in \mathbb{R}^{r\times d}$, if $A = U\Sigma V^\top$ is the singular value decomposition (SVD) of A, the projection of A onto $\mathrm{St}(d, r)$ can be obtained as follows:

$$\mathrm{Proj}_{\mathrm{St}(d,r)}(A) = UV^\top. \tag{4}$$

In this work, the following product manifold is also used extensively and we give it a specific name:

$$\mathcal{M}_{\mathrm{PGO}}(r, n) \triangleq (\mathrm{St}(d, r) \times \mathbb{R}^r)^n \subset \mathbb{R}^{r\times(d+1)n}. \tag{5}$$

In our notation (5), we highlight the constants r and n and omit the constant d, as the latter is essentially fixed (i.e., $d \in \{2, 3\}$). An explicit characterization of the tangent and normal spaces of $\mathcal{M}_{\mathrm{PGO}}(r, n)$ with their corresponding orthogonal projection operators are provided in our technical report [1, Eq. (8)–(10)].

II. Related Work

A. Centralized Pose-Graph Optimization Algorithms

In prior work, Rosen *et al.* [13] developed SE-Sync, a state-of-the-art certifiably correct algorithm for PGO. SE-Sync is based upon a (convex) semidefinite relaxation that its authors prove admits a *unique, low-rank* minimizer providing an *exact, globally optimal* solution to the original PGO problem whenever the noise on the available measurements is not too large; moreover, in the (typical) case that exactness obtains, it is possible to *verify* this fact *a posteriori* [11], thereby *certifying* the correctness (optimality) of the recovered estimate. To solve the resulting semidefinite relaxation efficiently, SE-Sync employs the Riemannian Staircase [15], which leverages symmetric low-rank (Burer–Monteiro) factorization [18] to directly search for a symmetric low-rank factor of the SDP solution, and implements this low-dimensional search using the truncated-Newton *Riemannian trust-region* (RTR) method [17], [19]. This combination of *low-rank factorization* and *fast local search* (via truncated-Newton RTR) enables SE-Sync to recover *certifiably globally optimal* PGO solutions at speeds comparable to (and frequently significantly faster than) the standard state-of-the-art *local* search methods (e.g., Gauss–Newton) [13].

Unfortunately, the specific computational synthesis proposed in SE-Sync does not admit a straightforward distributed implementation. In particular, the semidefinite relaxation underlying SE-Sync is obtained after exploiting the separable structure of PGO to *analytically eliminate* the translational variables [20]; while this has the advantage of reducing the problem to a lower-dimensional optimization over a *compact* search space, it also means that the objective matrix of the resulting SDP is generically *fully dense*, thereby leading to *loss of privacy* and incurring a significant communication cost (see Remark 1).

A similar centralized solver, Cartan-Sync, is proposed by Briales and Gonzalez-Jimenez [14]. The main difference between SE-Sync and Cartan-Sync is that the latter directly relaxes the PGO problem *without* first analytically eliminating the translations; consequently, the resulting relaxation retains the sparsity present in the original PGO problem. However, this alternative SDP relaxation (and consequently Cartan-Sync itself) has *not* previously been shown to enjoy any exactness guarantees; in particular, its minimizers, and their relation to solutions of PGO, have not previously been characterized. As one of the main contributions of this work, we derive sharp correspondences between minimizers of Cartan-Sync's relaxation and the original relaxation employed by SE-Sync (see Theorem 1); in particular, this correspondence enables us to *extend* the exactness guarantees of the latter to cover the former, thereby justifying its use as a basis for distributed certifiably correct PGO algorithms.

As a related note, similar SDP relaxations [21]–[24] have also been proposed for *rotation averaging* [25]. This problem arises in a number of important applications such as camera network localization (CNL) [26]–[28], structure from motion [25], and other domains such as cryo-electron microscopy in structural biology [29]. Mathematically, rotation averaging can be derived as a *specialization* of PGO obtained by setting the measurement precisions for the translational observations to 0 (practically, *ignoring* translational observations and states); consequently, the algorithms proposed in this work immediately apply, *a fortiori*, to distributed rotation averaging.

B. Decentralized Pose-Graph Optimization Algorithms

The work by Choudhary *et al.* [6] is currently the state of the art in distributed PGO solvers and has been recently used by modern decentralized CSLAM systems [30], [31]. Choudhary *et al.* [6] propose a two-stage approach for finding approximate solutions to PGO. The first stage approximately solves the

underlying rotation averaging problem by relaxing the nonconvex $\mathrm{SO}(d)$ constraints, solving the resulting (unconstrained) linear least-squares problem, and projecting the results back to $\mathrm{SO}(d)$. The rotation estimates are then used in the second stage to initialize a single Gauss–Newton iteration on the full PGO problem. In both stages, iterative and distributable linear solvers such as Jacobi over-relaxation (JOR) and successive over-relaxation (SOR) [32] are used to solve the normal equations. The experimental evaluations presented in [6] demonstrate that this approach significantly outperforms prior techniques [8], [9]. Nonetheless, the proposed approach is still performing incomplete local search on a nonconvex problem and, thus, cannot offer any performance guarantees.

In another line of work, Tron *et al.* [26]–[28] propose a multistage distributed Riemannian consensus protocol for CNL based on distributed execution of Riemannian gradient descent (RGD) over $\mathcal{M}^n$ where $\mathcal{M} = \mathrm{SO}(3) \times \mathbb{R}^3$ and n is the number of cameras (agents). CNL can be seen as a special instance of collaborative PGO where each agent owns a *single* pose rather than an entire trajectory. In these works, the authors establish convergence to critical points and, under *perfect* (noiseless) measurements, convergence to globally optimal solutions. See [1, Remark 6] for a specialized form of our algorithm suitable for solving CNL.

Recently, Fan and Murphy [7], [10] proposed a majorization–minimization approach to solve distributed PGO. Each iteration constructs a quadratic upper bound on the cost function, and minimization of this upper bound is carried out in a distributed and parallel fashion. The core benefits of this approach are that it is *guaranteed* to converge to a first-order critical point of the PGO problem, and that it allows one to incorporate Nesterov's acceleration technique, which provides significant empirical speedup on typical PGO problems. In this work, we propose a different local search method that performs BCD over Riemannian manifolds. Similar to [7] and [10], we achieve significant empirical speedup by adapting Nesterov's accelerated coordinate descent scheme [33] while ensuring global convergence using adaptive restart [34]. However, compared to [7], our method enjoys the computational advantage that each iteration requires only an inexpensive *approximate* solution of each local subproblem (as opposed to fully solving the local subproblem to a first-order critical point). Lastly, while all distributed methods [6], [7], [10], [26]–[28] reviewed, thus, far are *local* search techniques, our approach is the first *global* and *certifiably correct* distributed PGO algorithm.

C. Block-Coordinate Descent Methods

BCD methods (also known as Gauss–Seidel-type methods) are classical techniques [32], [35] that have recently regained popularity in large-scale machine learning and numerical optimization [36]–[39]. These methods are popular due to their simplicity, cheap iterations, and flexibility in the parallel and distributed settings [32].

BCD is a natural choice for solving PGO in the distributed setting due to the graphical decomposition of the underlying optimization problem. In fact, BCD-type techniques have been applied in the past [40], [41] to solve SLAM. Similarly, in computer vision, variants of the Weiszfeld algorithm have also been used for robust rotation averaging [25], [42]. The abovementioned works, however, use BCD for *local* search and, thus, cannot guarantee global optimality. More recently, Eriksson *et al.* [23] propose the row-by-row (RBR) method, a BCD-type algorithm for solving the SDP relaxation of rotation averaging.

RBR needs to store and manipulate a *dense* $dn \times dn$ matrix, which is not scalable to SLAM applications (where n is usually one to two orders of magnitude larger than the problems considered in [23]). Furthermore, although in principle RBR can be executed distributedly, it requires each agent to estimate a block-row of the SDP variable, which in the case of rotation averaging corresponds to both public and private rotations of other agents. In contrast, the method proposed in this work does not leak information about robots' private poses throughout the optimization process.

This work is originally inspired by recent block-coordinate minimization algorithms for solving SDPs with diagonal constraints via the Burer–Monteiro approach [43], [44]. Our recent technical report [45] extends these algorithms and the global convergence rate analysis provided by Erdogdu *et al.* [44] from the unit sphere (SDPs with diagonal constraints) to the Stiefel manifold (SDPs with *block*-diagonal constraints). In this work, we further extend our initial results by providing a unified Riemannian BCD algorithm and its global convergence rate analysis.

III. Certifiably Correct Pose-Graph Optimization

In this section, we formally introduce the PGO problem, its semidefinite relaxation, and our certifiably correct algorithm for solving these in the distributed setting. Fig. 1 summarizes the problems introduced in this section and how they relate to each other.

A. Pose-Graph Optimization

PGO is the problem of estimating unknown poses from noisy relative measurements. PGO can be modeled as a directed graph (a *pose graph*) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = [n]$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ correspond to the sets of unknown poses and relative measurements, respectively. In the rest of this article, we make the standard assumption that $\mathcal{G}$ is weakly connected. Let $T_1, T_2, \ldots, T_n \in \mathrm{SE}(d)$ denote the poses that need to be estimated, where each $T_i = (R_i, t_i)$ consists of a rotation component $R_i \in \mathrm{SO}(d)$ and a translation component $t_i \in \mathbb{R}^d$. Following [13], we assume that for each edge $(i, j) \in \mathcal{E}$, the corresponding relative measurement $\widetilde{T}_{ij} = (\widetilde{R}_{ij}, \widetilde{t}_{ij})$ from pose T_i to T_j is generated according to the following:

$$\widetilde{R}_{ij} = \underline{R}_{ij} R^{\epsilon}_{ij},\ R^{\epsilon}_{ij} \sim \mathrm{Langevin}(I_d, \kappa_{ij}) \tag{6}$$

$$\widetilde{t}_{ij} = \underline{t}_{ij} + t^{\epsilon}_{ij},\ t^{\epsilon}_{ij} \sim \mathcal{N}(0, \tau_{ij}^{-1} I_d). \tag{7}$$

In the abovementioned equation, $\underline{R}_{ij} \triangleq \underline{R}_i^{\top} \underline{R}_j$ and $\underline{t}_{ij} \triangleq \underline{R}_i^{\top} (\underline{t}_j - \underline{t}_i)$ denote the true (noiseless) relative rotation and translation, respectively. Under the noise model (6)–(7), it can

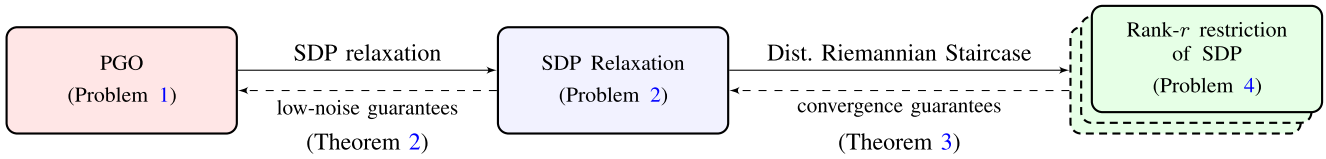


Fig. 1. Relations between problems considered in this work. From the MLE formulation of PGO (see Problem 1), applying semidefinite relaxation yields the SDP (see Problem 2). Applying a distributed implementation of the Riemannian staircase algorithm [15] and BM factorization [18] on the SDP then yields a set of rank-restricted problems (see Problem 4), which can be *locally* optimized using our distributed Riemannian local search method (see Section IV). After local search, the *global* optimality of the recovered first-order critical points of Problem 4 as solutions of the original SDP (see Problem 2) can then be checked via *posthoc* verification, and if necessary, a descent direction can be constructed from a suboptimal critical point to continue the search (see Section V). Finally, under sufficiently low noise, SDP relaxations are guaranteed to find *global* minimizers of PGO (see Theorem 2).

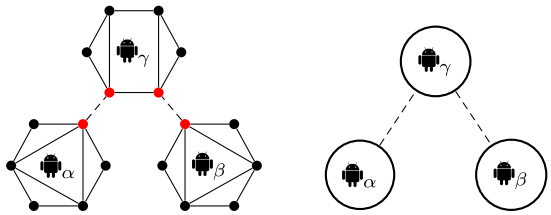


Fig. 2. Collaborative PGO in CSLAM. (a) Multiple robots must jointly estimate their trajectories in the same frame. Each robot has multiple pose variables that are connected by odometry measurements and loop closures. We refer to poses that have interrobot loop closures (dashed edges) as *public* (marked in red), and all other poses as *private* (marked in black). (b) Dependence graph for the CSLAM pose graph shown in (a). Each vertex corresponds to a robot, and two vertices are adjacent if and only if there exists at least one interrobot loop closure between the corresponding robots.

be shown that a maximum likelihood estimate (MLE) is obtained as a minimizer of the following nonconvex optimization problem [13].

Problem 1 (Pose-Graph Optimization):

$$\underset{\{R_i\},\{t_i\}}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \kappa_{ij}\|R_j - R_i\widetilde{R}_{ij}\|_F^2 + \tau_{ij}\|t_j - t_i - R_i\widetilde{t}_{ij}\|_2^2, \tag{8a}$$

$$\text{subject to} \quad R_i \in \mathrm{SO}(d),\, t_i \in \mathbb{R}^d \quad \forall i \in [n]. \tag{8b}$$

Collaborative PGO: In collaborative PGO, multiple robots must collaboratively estimate their trajectories in a common reference frame by solving the *collective* PGO problem *distributedly* (i.e., without outsourcing data to a single "central" node) using interrobot collaboration. Each vertex of the collective pose graph represents the pose of a robot at a certain time step. Odometry measurements and *intrarobot* loop closures connect poses within a single robot's trajectory. When two robots visit the same place (not necessarily at the same time), they establish *interrobot* loop closures that link their respective poses; see [30], [46], [47], and references therein for resource-efficient distributed interrobot loop closure detection techniques. Fig. 2 illustrates a simple example based on CSLAM. Interrobot loop closures induce a natural partitioning of collective pose graph nodes into public and private poses, marked in red and black in Fig. 2, respectively. Additionally, these interagent measurements also create dependencies between the robots. This is captured by the *dependence graph* shown in Fig. 2(b).

Definition 1 (Public and private poses): Poses that share interrobot loop closures with poses of other robots are called *public* poses (or *separators* [8]). All other poses are *private* poses.

B. SDP Relaxation for PGO

Traditionally, Problem 1 is solved with local search algorithms such as Gauss–Newton. However, depending on the noise level and the quality of initialization, local search algorithms are susceptible to local minima [11]. To address this critical issue, recent works aim to develop *certifiably correct* PGO solvers. In particular, techniques based on SDP relaxation demonstrate empirical state-of-the-art performance while providing theoretical correctness (global optimality) guarantees under low noise regimes [13], [22], [23].

In this section, we present a semidefinite relaxation of Problem 1 that was first studied in [14]. Let $T \triangleq [R_1\ t_1\ \ldots\ R_n\ t_n] \in (\mathrm{SO}(d) \times \mathbb{R}^d)^n$ be the block-row matrix obtained by aggregating all rotation and translation variables. Briales and Gonzalez-Jimenez [14] show that the cost function (8a) in Problem 1 can be written in the matrix form as $f(T) = \langle Q, T^\top T\rangle$, where $Q \in \mathbb{S}^{(d+1)n}$ is a symmetric matrix known as the *connection Laplacian* formed using all relative measurements. Consider the "lifted" variable $Z = T^\top T \in \mathbb{S}_+^{(d+1)n}$. Treating Z as a $(d+1)\times(d+1)$-block-structured matrix, we see that several necessary conditions for Z to satisfy the original constraints (8b) in PGO are as follows:

$$Z \succeq 0 \tag{9}$$

$$\mathrm{rank}(Z) = d \tag{10}$$

$$Z_{[i,i](1:d,1:d)} = R_i^\top R_i = I_{d\times d} \quad \forall i \in [n] \tag{11}$$

$$\det(Z_{[i,j](1:d,1:d)}) = \det(R_i^\top R_j) = 1 \quad \forall i,j \in [n], i \neq j. \tag{12}$$

Dropping the nonconvex rank and determinant constraints (10) and (12) yields an SDP relaxation of Problem 1.

Problem 2 (SDP Relaxation for PGO [14]):

$$\underset{Z\in\mathbb{S}_+^{n+dn}}{\text{minimize}} \quad \langle Q, Z\rangle \tag{13a}$$

$$\text{subject to} \quad Z_{[i,i](1:d,1:d)} = I_{d\times d} \quad \forall i \in [n]. \tag{13b}$$

The original SE-Sync algorithm [13] employs a different SDP relaxation for Problem 1, obtained by first exploiting the so-called *separable* structure of PGO [20] to analytically eliminate the translation variables, and *then* performing convex relaxation over the resulting rotation-only problem. This approach yields the following.

Problem 3 (Rotation-only SDP Relaxation for PGO [13]):

$$\underset{Z_{\mathrm{R}} \in \mathbb{S}^{dn}_{+}}{\text{minimize}} \quad \langle Q_{\mathrm{R}}, Z_{\mathrm{R}} \rangle \tag{14a}$$

$$\text{subject to} \quad Z_{\mathrm{R}[i,i]} = I_{d\times d} \quad \forall i \in [n] \tag{14b}$$

where Q_{R} is obtained by computing a generalized Schur complement of the connection Laplacian Q [1, App. A].

Remark 1 (Choosing the right SDP for distributed PGO): Problem 3 has several advantages over Problem 2, including a compact search space and better numerical conditioning. Nevertheless, unlike Q in Problem 2, the cost matrix Q_{R} in Problem 3 is generally *dense* [1, App. A.1]. In graphical terms, eliminating the translation variables makes the underlying dependence graph *fully connected*. This is a major drawback in the distributed setting, since it corresponds to making all of the poses *public*, thereby substantially increasing the required communication. As we shall see in the following sections, our proposed algorithm relies on and exploits the *sparse* graphical structure (both intrarobot and interrobot) of the problem to achieve computational and communication efficiency, and to preserve the privacy of participating robots. Therefore, in this work we seek to solve Problem 2 as a sparse convex relaxation to PGO.

However, in contrast to the SE-Sync relaxation (see Problem 3) [13], Problem 2 has *not* previously been shown to enjoy any exactness guarantees. We now present new results to characterize the connection between the solutions of these problems, thereby *extending* the guarantee of exactness from Problem 3 to Problem 2.

Theorem 1 (Connection between Problems 2 and 3): Problem 2 admits a minimizer $Z^\star$ with $\mathrm{rank}(Z^\star) = r$ *if and only if* Problem 3 admits a minimizer $Z^\star_{\mathrm{R}}$ with the same rank. Furthermore, $\langle Q, Z^\star \rangle = \langle Q_{\mathrm{R}}, Z^\star_{\mathrm{R}} \rangle$ for *all* minimizers of Problem 2 and Problem 3.

A proof is provided in [1, App. A]. Theorem 1 indicates that relaxing the additional translational variables when forming Problem 2 does not weaken the relaxation versus the SE-Sync relaxation (see Problem 3), *nor* (crucially) introduce any additional minimizers that do *not* correspond to PGO solutions. In particular, Theorem 1 and [13, Proposition 2] together imply the following *exactness* guarantee for Problem 2 under low measurement noise.

Theorem 2 (Exact recovery via Problem 2): Let $\underline{Q}$ be the connection Laplacian in Problem 2, constructed using the true (latent) relative transformations $(\underline{R}_{ij}, \underline{t}_{ij})$. There exists a constant $\delta > 0$ such that if $\|Q - \underline{Q}\|_2 < \delta$, every minimizer $Z^\star$ to Problem 2 has its first $d \times (n + dn)$ block row given by

$$Z^\star{}_{(1:d,:)} = \begin{bmatrix} R^\star_1 & t^\star_1 & \dots & R^\star_n & t^\star_n \end{bmatrix} \tag{15}$$

where $\{R^\star_i, t^\star_i\}$ is an optimal solution to Problem 1.

A proof is provided in [1, App. A]. Theorem 2 provides a crucial missing piece for achieving certifiably correct *distributed* PGO solvers: under low noise (quantified by the deviation in spectral norm of the connection Laplacian Q from its latent value), one can directly read off a global minimizer to PGO (see Problem 1) from the first block row of any solution $Z^\star$ of the sparse SDP relaxation (see Problem 2). As empirically shown in [13] and [14], both SDP relaxations are exact in real-world scenarios (see Section VII for additional empirical evidence).

C. Solving SDP: The Distributed Riemannian Staircase

In typical CSLAM scenarios, the dimension of the SDP relaxation can be quite large (e.g., $\dim(Z) > 10^4$), and thus, it is often impractical to solve Problem 2 using standard (interior-point) methods. In a seminal paper, Burer and Monteiro[18] proposed a more scalable approach to search for *low-rank* solutions $Z^\star$ in particular: *assume* that some solution admits a symmetric low-rank factorization of the form $Z^\star = X^{\star\top} X^\star$, where $X^\star \in \mathbb{R}^{r\times n}$ and $r \ll n$, and then directly search for the low-rank factor $X^\star$. This substitution has the two-fold effect of, first, dramatically reducing the dimension of the search space, and, second, rendering the positive semidefiniteness constraint on Z *redundant*, since $X^\top X \succeq 0$ for *any* $X \in \mathbb{R}^{r\times n}$. In consequence, the *rank-restricted* version of the original semidefinite program obtained by performing the substitution $Z = X^\top X$ is actually a lower-dimensional *nonlinear program*, and so can be processed much more efficiently using standard (local) NLP methods.

For SDPs with block-diagonal constraints, Boumal [15] extends the general approach of Burer and Monteiro [18] by further exploiting the *geometric* structure of the constraints in the Burer–Monteiro-factored (BM) problem. The result is an elegant algorithm known as the *Riemannian Staircase*, which is used to solve the (large-scale) semidefinite relaxations in SE-Sync [13] and Cartan-Sync [14].

In this work, we show how to implement the Riemannian Staircase approach in a *distributed* manner, thereby enabling us to solve collaborative PGO. Algorithm 1 presents our distributed Riemannian Staircase algorithm. In each iteration of the Riemannian Staircase, we assume a symmetric rank-r factorization $Z = X^\top X$ where $X \in \mathbb{R}^{r\times(n+dn)}$. Writing the blocks of X as $X = [Y_1\ p_1\ \dots\ Y_n\ p_n]$, the SDP constraints (13b) require that $Y_i \in \mathrm{St}(d, r)$ and $p_i \in \mathbb{R}^r$, for all $i \in [n]$. Equivalently, the aggregate variable X is constrained to live on the product manifold $\mathcal{M}_{\mathrm{PGO}}(r, n) \triangleq (\mathrm{St}(d, r) \times \mathbb{R}^r)^n$. Imposing the rank-$r$ factorization, thus, transforms the original SDP into the following *rank-restricted* problem:

Problem 4 (Rank-restricted SDP for PGO):

$$\underset{X \in \mathcal{M}_{\mathrm{PGO}}(r,n)}{\text{minimize}} \quad \langle Q, X^\top X \rangle. \tag{16}$$

In Section IV, we develop a novel local search algorithm, RBCD, which we will use to recover first-order critical points of the rank-restricted SDP (see Problem 4) in the distributed setting. Inspired by Nesterov's accelerated coordinate descent [33], we also propose an accelerated variant, RBCD++. We also establish global first-order convergence guarantees for both RBCD and RBCD++ (see Theorem 4 and [1, Sec. 5]).

Algorithm 1: Distributed Riemannian Staircase.

Input: - Initial point $X \in \mathcal{M}_{\text{PGO}}(r_0, n)$.
Output: - A minimizer $X^\star \in \mathcal{M}_{\text{PGO}}(r, n)$ of Problem 4 such that $Z^\star = (X^\star)^\top (X^\star)$ is a solution to Problem 2.
1: **for** $r = r_0, r_0 + 1, \ldots$ **do**
2: Compute first-order critical point of Problem 4 (Section IV)

$$X^\star \leftarrow \text{RBCD}(X)$$

3: Lift to first-order critical point at next level as in (35)

$$X^\star \leftarrow [(X^\star)^\top \; 0]^\top$$

4: Construct corresponding dual certificate matrix $S(X^\star)$ in (34).
5: Compute minimum eigenpair (Algorithm 6)

$$(\lambda, v) \leftarrow \text{MinEigS}(\text{X}^\star)$$

6: **if** $\lambda \geq 0$ **then**
7: **Return** $X^\star$.
8: **else**
9: Construct second-order descent direction $\dot{X}_+$ in (36).
10: Descend from $X^\star$ (Algorithm 7)

$$X \leftarrow \textsc{EscapeSaddle}(X^\star, \dot{X}_+)$$

11: **end if**
12: **end for**

From a first-order critical point $X^\star$, we ultimately wish to recover a solution to the SDP relaxation. To do so, we first lift $X^\star$ to the next level of the staircase (i.e., increment rank r by one). This can be trivially done by padding $X^\star$ with a row of zeros (see line 1). The motivations behind this operation will become clear later. By construction, the matrix $Z = X^{\star\top} X^\star$ is feasible in Problem 2. We may verify the global optimality of Z by checking the (necessary and sufficient) Karush–Kuhn–Tucker (KKT) conditions; for a first-order critical point $X^\star$, this amounts to verifying that a certain dual certificate matrix $S(X^\star)$ is positive semidefinite (see line 1). In Section V, we present the first distributed procedure to carry out this verification. If the dual certificate has a negative eigenvalue, then Z is *not* a minimizer of the SDP, and $X^\star$ is in fact a *saddle point* to Problem 4. Fortunately, in this case, the procedure in Section V also returns a descent direction, with which we can escape the saddle point (see line 1) and restart distributed local search.

Remark 2 (First-order versus second-order optimization in the Riemannian Staircase): The formulation of the Riemannian Staircase presented in Algorithm 1 differs *slightly* from its original presentation in [15]: specifically, the latter presupposes access to an algorithm that is capable of computing *second*-order critical points of Problem 4, whereas the RBCD method we employ in line 1 only guarantees convergence to *first*-order critical points. This has implications for the convergence properties of the overall algorithm: while one can show that the second-order version of the Riemannian Staircase [15, Algorithm 1] is guaranteed to terminate at a level $r \leq n$ when applied to Problem 2 [15, Th. 3.8],[1] the weaker (first-order) guarantees provided by RBCD are reflected in a correspondingly weaker set of convergence guarantees for our (first-order) Algorithm 1 provided in the following theorem.

Theorem 3 (Convergence of Algorithm 1): Let $\{X^{(r)}\}$ denote the sequence of low-rank factors generated by Algorithm 1 in line 1 using a particular saddle escape procedure described in [1, App. C]. Then exactly one of the following two cases holds.

i) Algorithm 1 terminates after finitely many iterations and returns a symmetric factor $X^{(r)}$ for a minimizer $Z^\star = (X^{(r)})^\top X^{(r)}$ of Problem 2 in line 1.
ii) Algorithm 1 generates an infinite sequence $\{X^{(r)}\}$ satisfying $f(X^{(r_2)}) < f(X^{(r_1)})$ for all $r_2 > r_1$, with

$$\lim_{r\to\infty} f(X^{(r)}) = f^\star_{\text{SDP}} \tag{17}$$

and there exists an infinite subsequence $\{X^{(r_k)}\} \subset \{X^{(r)}\}$ satisfying:

$$\lim_{k\to\infty} \lambda_{\min}\left(S(X^{(r_k)})\right) = 0. \tag{18}$$

In a nutshell, Theorem 3 states that Algorithm 1—with a particular version of saddle escape procedure described in [1, App. C]—either terminates after a finite number of iterations, or generates an infinite sequence of factors $\{X^{(r)}\}$ that monotonically strictly decrease the objective to the optimal value $f^\star_{\text{SDP}}$ *and* that can arbitrarily well-approximate the satisfaction of the KKT condition $\lambda_{\min}(S(X^{(r)})) \geq 0$. We prove this theorem in [1, App. C].

We remark that while the convergence guarantees of Theorem 3 are *formally* weaker than those achievable using a second-order local search method, as a *practical* matter these differences are inconsequential. In any numerical implementation of the Riemannian Staircase framework, both the second-order criticality of a stationary point (in the second-order version) and the nonnegativity of the minimum eigenvalue λ (in Algorithm 1) are checked subject to some numerical tolerance $\epsilon_{\text{tol}} > 0$; this accounts for both the finite precision of real-world computers, and the fact that the low-rank factors X computed via local search in line 1 are *themselves* only *approximations* to critical points, as they are obtained using iterative local optimization methods. In particular, practical implementations of Algorithm 1 (including ours) would replace line 1 with a termination condition of the form "$\lambda \geq -\epsilon_{\text{tol}}$,"[2] and (18) guarantees that this condition is satisfied after *finitely* many iterations for *any* $\epsilon_{\text{tol}} > 0$. As a practical matter, the behavior of Algorithm 1 is far from the pessimistic case described in part (ii) of Theorem 3; as we show empirically in Section VII, in real-world applications typically only 1–3 iterations suffice.

[1] Strictly speaking, the finite-termination guarantees provided by [15, Theorem 3.8] only hold for the SE-Sync relaxation Problem 3 (cf. [13, Proposition 3]); however, we can extend these guarantees to Problem 2 by exploiting the correspondence between critical points of the low-rank factorizations of Problems 2 and 3 that we establish in [1, Lemma 3 in App. A].

[2] This is analogous to the standard stopping criterion $\|\nabla f(x)\| < \epsilon_{\text{tol}}$ for local optimization methods.

Algorithm 2: Distributed Certifiably Correct Pose Graph Optimization (DC2-PGO)

Input: - Initial rank $r_0 \geq d$ for the Riemannian Staircase.

Output: - A feasible solution $T \in \mathrm{SE}(d)^n$ to Problem 1 and the lower bound $f^\star_{\mathrm{SDP}}$ on Problem 1's optimal value.

1: Obtain initial point $X \in \mathcal{M}_{\mathrm{PGO}}(r, n)$ via distributed initialization.
2: $X^\star \leftarrow \mathrm{DistributedRiemannianStaircase}(X)$.
3: Recover optimal value of the SDP relaxation
$$f^\star_{\mathrm{SDP}} = \langle Q, X^{\star\top} X^\star \rangle$$
4: From $X^\star$, obtain feasible $T \in \mathrm{SE}(d)^n$ via distributed rounding.
5: **Return** $T, f^\star_{\mathrm{SDP}}$.

D. Complete Algorithm

The distributed Riemannian Staircase (see Algorithm 1) is the core computational procedure of our overall algorithm. Nevertheless, to implement a *complete* distributed method for solving the original PGO problem (see Problem 1), we must still specify procedures for, first, initializing the Riemannian Staircase by constructing an initial point $X \in \mathcal{M}_{\mathrm{PGO}}(r_0, n)$, and, second, *rounding* the low-rank factor $X^\star$ returned by the Riemannian Staircase to extract a feasible solution $T \in \mathrm{SE}(d)^n$ of the PGO problem. We discuss the details of both distributed initialization and rounding in Section VI. Combining these procedures produces our complete distributed certifiably correct algorithm, DC2-PGO (see Algorithm 2).

Since the SDP (see Problem 2) is a convex relaxation of PGO, its optimal value $f^\star_{\mathrm{SDP}}$ is necessarily a lower bound on the global minimum of PGO. Using this fact, we may obtain an *upper* bound on the suboptimality of the solution returned by DC2-PGO. Specifically, let $f(T)$ denote the objective achieved by the final estimate, and let $f^\star_{\mathrm{MLE}}$ denote the optimal value of Problem 1. Then:

$$f(T) - f^\star_{\mathrm{MLE}} \leq f(T) - f^\star_{\mathrm{SDP}}. \tag{19}$$

In particular, if $f(T) = f^\star_{\mathrm{SDP}}$, then the SDP relaxation is *exact*, and $f(T) = f^\star_{\mathrm{MLE}}$. In this case, (19) serves as a *certificate* of the *global* optimality of T.

IV. Distributed Local Search Via Riemannian Block-Coordinate Descent

In this section, we introduce a new *distributed* local search algorithm to identify a first-order critical point of the rank-restricted SDP relaxation (see Problem 4), which is needed by the Distributed Riemannian Staircase framework (see Algorithm 1, line 1). Our algorithm is applicable to a broad class of smooth optimization problems defined over the Cartesian product of Riemannian manifolds:

$$\underset{X \in \mathcal{M}}{\text{minimize}} \quad f(X), \quad \mathcal{M} \triangleq \mathcal{M}_1 \times \ldots \times \mathcal{M}_N. \tag{20}$$

The abovementioned problem contains Problem 4 as a special case. Specifically, in distributed PGO, each block b exactly corresponds to a robot and $\mathcal{M}_b = \mathcal{M}_{\mathrm{PGO}}(r, n_b) \triangleq (\mathrm{St}(d, r) \times \mathbb{R}^r)^{n_b}$ corresponds to the search space of this robot's trajectory. Here, n_b is the number of poses owned by the robot associated to block b, and N is the total number of robots. For this reason, unless otherwise mentioned, in this section, we use the words "block" and "robot" interchangeably.

To solve (20), we leverage the product structure of the underlying manifold and propose a distributed *BCD* algorithm that we call RBCD (see Algorithm 3). In each iteration of RBCD, a block $b \in [N]$ is selected to be optimized. Specifically, let $X_b \in \mathcal{M}_b$ be the component of X corresponding to the selected block, and let $\widehat{X}_{[N]\setminus\{b\}}$ be the (fixed) values of remaining blocks. We update X_b by minimizing the following *reduced cost function*:

$$\underset{X_b \in \mathcal{M}_b}{\text{minimize}} \quad f_b(X_b) \triangleq f(X_b, \widehat{X}_{[N]\setminus\{b\}}). \tag{21}$$

For the rank-restricted SDP (see Problem 4) in PGO, the reduced problem for block b takes the form

$$f_b(X_b) = \langle Q_b, X_b^\top X_b \rangle + 2\langle F_b, X_b \rangle + \text{const.} \tag{22}$$

In the abovementioned equation, Q_b is the submatrix of Q formed with the rows and columns that correspond to block b (i.e., the trajectory of robot b), and $F_b \in \mathbb{R}^{r \times (d+1)n_b}$ is a constant matrix that depends on the (fixed) public variables of robot bs neighbors in the pose graph.

Remark 3 (Communication requirements of RBCD): RBCD is designed such that it can be easily implemented by a network of robots. At each iteration, the team first coordinates to select the next block (robot) to update (see Algorithm 3, line 3). Then, to update the selected block (see Algorithm 3, line 3), the robot corresponding to this block receives public variables from its neighboring robots in the pose graph. Afterwards, this robot forms and solves its local optimization problem (21), which does not require further communications. Finally, to determine when to terminate RBCD (see Algorithm 3, line 3), robots need to collaboratively evaluate their total gradient norm. In practice, checking the termination condition may be done periodically (instead of after every iteration) to save communication resources.

Remark 4 (Block-coordinate minimization on product manifolds): Prior works (e.g., [43]–[45]) have proposed similar *block-coordinate minimization* (BCM) algorithms to solve low-rank factorizations of SDPs with diagonal or block-diagonal constraints. Our approach generalizes these methods in two major ways. First, while prior methods are explicitly designed for problems over the product of spheres [43], [44] or Stiefel manifolds [45], our algorithm is applicable to the product of *any* Riemannian manifolds. Second, prior works [43]–[45] require the cost function to have a certain quadratic form, so that exact minimization of each variable block admits a closed-form solution. In contrast, our algorithm does not seek to perform exact minimization, but instead computes an inexpensive *approximate* update that achieves a *sufficient reduction* of the objective (see Section IV-B). This makes our method more general and applicable to a much broader class of smooth cost functions that satisfy

Algorithm 3: Riemannian Block-Coordinate Descent (RBCD).

Input:
- Global cost function $f : \mathcal{M} \triangleq \mathcal{M}_1 \times \ldots \times \mathcal{M}_N \to \mathbb{R}$.
- Initial solution $X^0 \in \mathcal{M}$.
- Stopping condition on gradient norm ϵ.

Output:
- First-order critical point $X^\star$.

1: $k \leftarrow 0$.
2: **while** $\| \operatorname{grad} f(X^k)\| > \epsilon$
3: Select next block $b_k \in [N]$.
4: Update the selected block $X_{b_t}^{k+1} \leftarrow \textsc{BlockUpdate}(f_{b_k}, X_{b_k}^t)$.
5: Carry over all other blocks $X_{b'}^{k+1} = X_{b'}^k$, $\forall b' \neq b_k$.
6: $k \leftarrow k + 1$.
7: **end while**
8: **return** $X^\star = X^k$.

a Lipschitz-type condition. We discuss this point in greater detail in [1, Sec. 5].

A. Block Selection Rules

In this section, we describe three mechanisms for selecting which block to update at each iteration of RBCD (see Algorithm 3, line 3). We note that similar rules have been proposed in the past; see, e.g., [36], [44], and [48].

1) Uniform Sampling. The first rule is based on the idea of uniform sampling. At each iteration, each block $b \in [N]$ is selected with equal probability $p_b = 1/N$.
2) Importance Sampling. In practice, it is often the case that selecting certain blocks leads to significantly better performance compared to others [36]. Therefore, it is natural to assign these blocks higher weights during the sampling process. We refer to this block selection rule as *importance sampling*. In this work, we set the probability of selecting each block to be proportional to the squared gradient norm, i.e., $p_b \propto \| \operatorname{grad}_b f(X)\|^2, \forall b \in [N]$. Here, $\operatorname{grad}_b f(X)$ denotes the component of the Riemannian gradient of $f(X)$ that corresponds to block b. Under Lipschitz-type conditions, the squared gradient norm can be used to construct a lower bound on the achieved cost decrement; see [1, Lemma 3].
3) Greedy [Gauss–Southwell (GS)]. We can also modify importance sampling into a deterministic strategy that simply selects the block with the largest squared gradient norm, i.e., $b \in \arg\max \| \operatorname{grad}_b f(X)\|^2$. We refer to this strategy as *greedy selection* or the *GS* rule [36]. Recent works also propose other variants of greedy selection such as Gauss–Southwell–Lipschitz (GSL) and Gauss–Southwell-Quadratic (GSQ) [36]. However, such rules require additional knowledge about the block Lipschitz constants that are hard to obtain in our application. For this reason, we restrict our deterministic selection rule to GS. Despite its simplicity, empirically the GS rule exhibits satisfactory performance; see Section VII.

Algorithm 4: BlockUpdate.

Input:
- Reduced cost function $f_b : \mathcal{M}_b \to \mathbb{R}$.
- Current block estimate $X_b^k \in \mathcal{M}_b$.
- User-specified mapping on tangent space $H : T_{X_b^k} \to T_{X_b^k}$ (default to Riemannian Hessian).
- Initial trust-region radius Δ_0.

Output:
- Updated block estimate $X_b^{k+1} \in \mathcal{M}_b$.

1: $\Delta \leftarrow \Delta_0$.
2: Form model function $\hat{m}_b(\eta_b)$ according to (24).
3: **while** true **do**
4: Compute an approximate solution $\eta_b^\star \in T_{X_b^k}\mathcal{M}_b$ to the trust-region subproblem (25), such that $\eta_b^\star$ produces at least a fixed fraction of the Cauchy decrease on the model function.
5: **if** $\rho(\eta_b^\star) > 1/4$ **then**
6: **return** $X_b^{k+1} = \operatorname{Retr}_{X_b^k}(\eta_b^\star)$.
7: **else**
8: Decrease trust-region radius $\Delta \leftarrow \Delta/4$.
9: **end if**
10: **end while**

Remark 5 (Communication requirements of different block selection rules): In practice, uniform sampling does not incur communication overhead, and can be approximately implemented using synchronized clocks on each robot (to conduct and coordinate BCD rounds) and a common random seed for the pseudorandom number generator (to agree on which robot should update in the next round). In contrast, importance sampling and greedy selection require additional communication overhead at each round, as robots need to evaluate and exchange local gradient norms. In particular, the greedy selection rule can be implemented via flooding gradient norms; see, e.g., the FloodMax algorithm for leader election in general synchronized networks [49, Ch. 4]. This requires robots to have unique IDs and communicate in synchronized rounds. While greedy and importance rules have higher communication overhead than uniform sampling, they also produce more effective iterations and, thus, converge faster (see Section VII).

B. Computing a Block Update

Note that since (21) is in general a nonconvex minimization, computing a block update by *exactly* solving this problem is intractable. In this section, we describe how to implement a cheaper approach that permits the use of *approximate* solutions of (21) by requiring only that they produce a *sufficient decrease* of the objective. While there are many options to achieve sufficient descent, in this work, we propose to (approximately) solve a single trust-region subproblem [17], [19]. Compared to a full minimization that would solve (21) to first-order critical point, our approach greatly reduces the computational cost. On the other hand, unlike other approximate update methods such as RGD, our method allows us to leverage (local) second-order

information of the reduced cost, which leads to more effective updates.

Let b be the block that we select to update, and denote the current value of this block (at iteration k) as X_b^k. We define the *pullback* of the reduced cost function (21) as follows [17], [50]:

$$\widehat{f}_b : T_{X_b^k} \to \mathbb{R} : \eta_b \mapsto f_b \circ \mathrm{Retr}_{X_b^k}(\eta_b). \tag{23}$$

Note that the pullback is conveniently defined on the tangent space, which itself is a vector space. However, since directly minimizing the pullback is nontrivial, it is approximated with a quadratic *model* function [17], [50], as defined below.

$$\widehat{m}_b(\eta_b) \triangleq f_b(X_b^k) + \langle \mathrm{grad}\, f_b(X_b^k), \eta_b \rangle + \frac{1}{2}\langle \eta_b, H[\eta_b] \rangle. \tag{24}$$

In (24), $H : T_{X_b^k} \to T_{X_b^k}$ is a user-specified mapping on the tangent space. By default, we use the Riemannian Hessian $H = \mathrm{Hess}\, f_b(X_b^k)$ so that the model function is a second-order approximation of the pullback. Then, we compute an update direction $\eta_b^\star$ on the tangent space by approximately solving the following trust-region subproblem:

$$\text{minimize}_{\eta_b \in T_{X_b^k}\mathcal{M}_b} \ \ \widehat{m}_b(\eta_b) \ \ \text{subject to} \ \ \|\eta_b\| \leq \Delta. \tag{25}$$

As a standard assumption, we require that the approximate solution must achieve at least as much decrease in the model function as a fixed fraction of the Cauchy step; see [1, Eq. (130)–(131)] for more details. In particular, the truncated conjugate-gradient (tCG) method [19] is guaranteed to satisfy this assumption. In our implementation, we use tCG together with a preconditioner similar to Cartan-Sync [14] for faster convergence.

To ensure that the obtained update direction yields sufficient descent on the original pullback, we follow the standard procedure [17], [19] and evaluate the following ratio that quantifies the *agreement* between model decrease (predicted reduction) and pullback decrease (actual reduction):

$$\rho(\eta_b^\star) \triangleq \frac{\widehat{f}_b(0) - \widehat{f}_b(\eta_b^\star)}{\widehat{m}_b(0) - \widehat{m}_b(\eta_b^\star)} = \frac{f_b(X_b^k) - \widehat{f}_b(\eta_b^\star)}{f_b(X_b^k) - \widehat{m}_b(\eta_b^\star)}. \tag{26}$$

If the abovementioned ratio is larger than a constant threshold (default to $1/4$), we accept the current update direction and set $X_b^{k+1} = \mathrm{Retr}_{X_b^k}(\eta_b^\star)$ as the updated value of this block. Otherwise, we reduce the trust-region radius Δ and solve the trust-region subproblem again. Algorithm 4 gives the pseudocode for the entire block update procedure. In the technical report [1, App. B], we prove that under mild conditions, the Cauchy step that serves as initialization to tCG is guaranteed to satisfy the required termination condition. Furthermore, the returned solution is guaranteed to produce *sufficient descent* on the cost function, which is crucial to establish global convergence rate of RBCD (see Algorithm 3).

Algorithm 4 aims at *sufficiently reducing* the cost function along a block coordinate. We note that in the special case of CNL where every block consists only of a *single* pose, one can perform *exact minimization* along any block coordinate. The reader is referred to [1, Remark 6] for further discussions.

Algorithm 5: Accelerated Riemannian Block-Coordinate Descent (RBCD++).

Input:
- Global cost function $f : \mathcal{M} \triangleq \mathcal{M}_1 \times \ldots \times \mathcal{M}_N \to \mathbb{R}$.
- Initial solution $X^0 \in \mathcal{M}$.
- Stopping condition on gradient norm ϵ.
- Restart constant $c_1 > 0$.

Output: - First-order critical point $X^\star$.

$k \leftarrow 0, V^0 \leftarrow X^0, \gamma_{-1} \leftarrow 0$.
while $\|\mathrm{grad}\, f(X^k)\| > \epsilon$ **do**
  $\gamma_k \leftarrow (1 + \sqrt{1 + 4\,N^2\gamma_{k-1}{}^2})/2\,N, \ \alpha_k \leftarrow 1/\gamma_k N$.
  // Y update
  $Y^k \leftarrow \mathrm{Proj}_{\mathcal{M}}((1-\alpha_k)X^k + \alpha_k V^k)$.
  // X update
  Select next block $b_k \in [N]$.
  Update the selected block $X_{b_k}^{k+1} \leftarrow \textsc{BlockUpdate}(f_{b_k}, Y_{b_k}^k)$.
  Carry over all other blocks $X_{b'}^{k+1} \leftarrow Y_{b'}^k, \forall b' \neq b_k$.
  // V update
  $V^{k+1} \leftarrow \mathrm{Proj}_{\mathcal{M}}(V^k + \gamma_k(X^{k+1} - Y^k))$.
  // Adaptive restart
  if $f(X^k) - f(X^{k+1}) < c_1 \|\mathrm{grad}_{b_k}\, f(X^k)\|^2$
  // Use default block update
  $X_{b_k}^{k+1} \leftarrow \textsc{BlockUpdate}(f_{b_k}, X_{b_k}^k)$.
  Carry over all other blocks $X_{b'}^{t+1} \leftarrow X_{b'}^k, \forall b' \neq b_k$.
  // Reset Nesterov's acceleration
  $V^{k+1} \leftarrow X^{k+1}$.
  $\gamma_k = 0$.
end if
$k \leftarrow k + 1$.
end while
return $X^\star = X^k$.

C. Accelerated Riemannian Block-Coordinate Descent

In practice, many PGO problems are poorly conditioned. Critically, this means that a generic first-order algorithm can suffer from slow convergence as the iterates approach a first-order critical point. Such slow convergence is also manifested by the typical *sublinear* convergence rate, e.g., for RGD, as shown in [50]. To address this issue, Fan and Murphy [7], [10] recently developed a majorization–minimization algorithm for PGO. Crucially, their approach can be augmented with a generalized version of Nesterov's acceleration that significantly speeds up empirical convergence.

Following the same vein of ideas, we show that it is possible to significantly speed up RBCD by adapting the celebrated accelerated coordinate-descent method (ACDM), originally developed by Nesterov [33] to solve smooth convex optimization problems. Compared to the standard randomized coordinate descent method, ACDM enjoys an accelerated convergence rate of $\mathcal{O}(1/k^2)$. Let N denote the dimension (number of coordinates) in the problem. ACDM updates two scalar sequences $\{\gamma_k\}, \{\alpha_k\}$ and three sequences of iterates $\{x^k\}, \{y^k\}, \{v^k\} \in \mathbb{R}^N$.

$$\gamma_k = (1 + \sqrt{1 + 4\,N^2 {\gamma_{k-1}}^2})/2\,N \tag{27}$$

$$\alpha_k = 1/(\gamma_k N) \tag{28}$$

$$y^k = (1 - \alpha_k)x^k + \alpha_k v^k \tag{29}$$

$$x^{k+1} = y^k - 1/L_{b_k} \nabla_{b_k} f(y^k) \tag{30}$$

$$v^{k+1} = v^k + \gamma_k (x^{k+1} - y^k). \tag{31}$$

In (30), L_{b_k} is the Lipschitz constant of the gradient that corresponds to coordinate b_k. Note that compared to standard references (e.g., [33] and [35]), we have slightly changed the presentation of ACDM, so that it can be extended to our Riemannian setting in a more straightforward manner. Still, it can be readily verified that (27)–(31) are equivalent to the original algorithm.[3]

In Algorithm 5, we adapt the ACDM iterations to design an accelerated variant of RBCD, which we call RBCD++. We leverage the fact that our manifolds of interest are naturally embedded within some linear space. This allows us to first perform the additions and subtractions as stated in (27)–(31) in the linear ambient space, and subsequently *project* the result back to the manifold. For our main manifold of interest $\mathcal{M}_{\text{PGO}}(r, n)$, the projection operation only requires computing the SVD for each Stiefel component, as shown in (4). Note that the original ACDM method performs a coordinate descent step (30) at each iteration. In RBCD++, we generalize (30) by employing the BLOCKUPDATE procedure (see Algorithm 4) to perform a descent step on a *block coordinate* $Y_b \in \mathcal{M}_b$ (see Algorithm 5, line 5).

Unlike the convex case, it is unclear how to prove convergence of the abovementioned acceleration scheme subject to the nonconvex manifold constraints. Fortunately, the convergence can be guaranteed by adding *adaptive restart* [34], which has also been employed in recent works [7], [10]. The underlying idea is to ensure that each RBCD++ update (specifically on the $\{X^t\}$ variables) yields a *sufficient reduction* of the overall cost function. This is quantified by comparing the descent with the squared gradient norm at the selected block (see Algorithm 5, line 5), where the constant $c_1 > 0$ specifies the minimum amount of descent enforced at each iteration. If this criterion is met, the algorithm simply continues to the next iteration. If not, the algorithm switches to the default block update method (same as RBCD), and restarts the acceleration scheme from scratch. Empirically, we observe that setting c_1 close to zero (corresponding to a permissive acceptance criterion) gives the best performance.

Remark 6 (Adaptive versus fixed restart schemes): Our adaptive restart scheme requires aggregating information from all robots to evaluate the cost function and gradient norm (see Algorithm 5, line 5). This step may become the communication bottleneck of the whole algorithm. While in theory, we need adaptive restart to guarantee convergence [1, Sec. 5], a practical remedy is to employ a *fixed restart* scheme [34] whereby we simply restart acceleration (see Algorithm 5, lines 5-5) periodically in fixed intervals. Our empirical results in Section VII show that the fixed restart scheme also achieves significant acceleration, although is inferior to adaptive restart scheme.

[3]For example, we can recover (27)–(31) from [35, Algorithm 4], by setting the strong convexity parameter σ to zero.

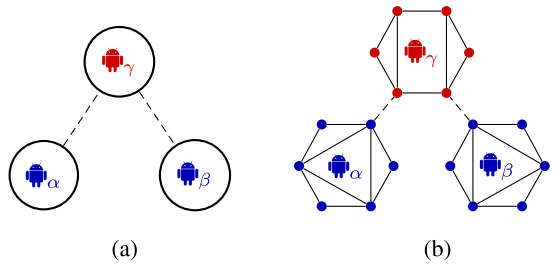


Fig. 3. Parallel updates for the collective pose graph shown in Fig. 2. (a) First, we find a coloring for the corresponding dependence graph such that adjacent robots have different colors. (b) Two coloring induces two "aggregate blocks" $\mathcal{A}_1, \mathcal{A}_2$ where $\mathcal{A}_1$ and $\mathcal{A}_2$ consist of all blue and red vertices, respectively. In each iteration, we select a color and update the corresponding variables. Note that $\mathcal{A}_1$ contains variables from both $_\alpha$ and $_\beta$, and therefore, these robots can update their variables in parallel when blue is selected. (a) Two-coloring for the dependence graph. (b) Induced aggregate blocks.

Remark 7 (Communication requirements of RBCD++): With fixed restart, the communication pattern of RBCD++ is identical to RBCD. In particular, with synchronized clocks, robots can update the scalars γ_k and α_k (see line 5) locally in parallel. Similarly, the "Y update" (see line 5) and "V update" steps (see line 5) do not require communication, since both only involve local linear combinations and projections to manifold. The main communication happens before the "X update" (see line 5), where each robot communicates the public components of their Y variables with neighbors in the global pose graph. Finally, if adaptive restart is used, robots need to communicate and aggregate the global cost and gradient norms to evaluate the restart condition (see line 5).

D. Parallel Riemannian Block-Coordinate Descent

Thus, far in each round of RBCD and RBCD++ (see Algorithms 3 and 5), exactly one robot performs BLOCKUPDATE (see Algorithm 4). However, after a slight modification of our blocking scheme, *multiple* robots may update their variables *in parallel* as long as they are *not* neighbors in the dependence graph [i.e., do not share an interrobot loop closure; see Fig. 2(b)]. This is achieved by leveraging the natural graphical decomposition of objectives in Problem 4 (inherited from Problem 1). Updating variables in parallel can significantly speed up the local search.

Gauss–Seidel-type updates can be executed in parallel using a classical technique known as red-black coloring (or, more generally, multicoloring schemes) [32]. We apply this technique to PGO (see Fig. 3):

1) First, we find a coloring for the set of *robots* such that *adjacent robots* in the dependence graph have different colors [see Fig. 3(a)]. Although finding a vertex coloring with the *smallest* number of colors is NP-hard, simple greedy approximation algorithms can produce a $(\Delta + 1)$-coloring, where Δ is the maximum degree of the dependence graph; see [51], [52], and the references therein for distributed algorithms. Note that Δ is often bounded

by a small constant due to the sparsity of the CSLAM dependence graph.

2) In each iteration, we select a color (instead of a single robot) by adapting the block selection rules presented in Section IV-A. The robots that have the selected color then update their variables in parallel.

Implementing the (generalized) importance sampling and greedy rules (see Section IV-A) with coloring requires additional coordination between the robots. In particular, the greedy rule requires computing the sum of squared gradient norms for each color at the beginning of each iteration. Similar to Section IV-A, a naïve approach would be to flood the network with the current squared gradient norms such that after a sufficient number of rounds (specifically, the diameter of the dependence graph), every robot aggregates all squared gradient norm information for every color. Robots can then independently compute the sum of squared gradient norms for every color and update their block only if their color has the largest gradient norm among all colors. We conclude this part by noting that it is also possible to allow *all* robots to update their *private* variables in *all* iterations (irrespective of the selected color) because private variables are separated from each other by public variables.

E. Convergence Rate Analysis and Guarantees

We conclude this section by establishing global convergence guarantees for RBCD (see Algorithm 3). In the technical report, we also establish similar convergence guarantees for RBCD++ (see Algorithm 5) [1, Theorem 5].

Theorem 4 (Global convergence rate of RBCD): Let $f^\star$ denote the global minimum of the optimization problem (20). Denote the iterates of RBCD (see Algorithm 3) as $X^0, X^1, \ldots, X^{K-1}$, and the corresponding block selected at each iteration as $b_0, \ldots, b_{K-1}$. Under mild assumptions specified in [1, Sec. 5], there exist block-specific constants $\lambda_b > 0$ such that RBCD with *uniform sampling* or *importance sampling* converges to a first-order critical point with the following rate:

$$\min_{0\le k\le K-1} \mathbb{E}_{b_{0:k-1}} \|\operatorname{grad} f(X^k)\|^2 \le \frac{4\,N(f(X^0)-f^\star)}{K\cdot \min_{b\in[N]}\lambda_b}. \quad (32)$$

In addition, RBCD with *greedy selection* converges with the following rate:

$$\min_{0\le k\le K-1} \|\operatorname{grad} f(X^k)\|^2 \le \frac{4N(f(X^0)-f^\star)}{K\cdot \min_{b\in[N]}\lambda_b}. \quad (33)$$

See [1, Sec. 5] for additional discussion and [1, App. B] for the proofs. While we focus on Problem 4, it is worth noting that the convergence guarantees derived in this section extend to *any* problem defined on product manifold, provided that they satisfy mild technical assumptions [1, Sec. 5]. Therefore, we believe that our algorithm and the convergence guarantees are also of interest to the broader optimization community.

V. Distributed Verification

In this section, we address the problem of *solution verification* [11] in the distributed setting. Concretely, we propose distributed solution verification and saddle escape algorithms to certify the optimality of a first-order critical point X of the rank-restricted relaxation (16) as a global minimizer $Z = X^\top X$ of problem (13), and for *escaping* from suboptimal stationary points after ascending to the next level of the Riemannian Staircase (see Algorithm 1). To the best of our knowledge, these are the first distributed solution verification algorithms to appear in the literature.

Our approach is based upon the following simple theorem of the alternative, which is a specialization of [53, Theorem 4] to problems (13) and (16):

Theorem 5 (Solution verification and saddle escape): Let $X \in \mathcal{M}_{\text{PGO}}(r,n)$ be a first-order critical point of the rank-restricted semidefinite relaxation (16), and define:

$$\Lambda(X) \triangleq \operatorname{SymBlockDiag}_d^+(X^\top X Q) \quad (34a)$$

$$S(X) \triangleq Q - \Lambda(X). \quad (34b)$$

Then exactly one of the following two cases holds.

a) $S(X) \succeq 0$ and $Z = X^\top X$ is a global minimizer of (13).

b) There exists $v \in \mathbb{R}^{(d+1)n}$ such that $v^\top S(X) v < 0$, and in that case

$$X_+ \triangleq \begin{bmatrix} X \\ 0 \end{bmatrix} \in \mathcal{M}_{\text{PGO}}(r+1,n) \quad (35)$$

is a first-order critical point of (16) attaining the same objective value as X, and

$$\dot{X}_+ \triangleq \begin{bmatrix} 0 \\ v^\top \end{bmatrix} \in T_{X_+}(\mathcal{M}_{\text{PGO}}(r+1,n)) \quad (36)$$

is a second-order direction of descent from X_+. In particular, taking v to be the eigenvector corresponding to the smallest eigenvalue of $S(X)$ satisfies the abovementioned conditions.

Remark 8 (Interpretation of Theorem 5): Let us provide a bit of intuition for what Theorem 5 conveys. Part (a) is simply the standard (necessary and sufficient) conditions for $Z = X^\top X$ to be the solution of the (convex) semidefinite program (13) [54]. In the event that these conditions are *not* satisfied [and, therefore, Z is *not* optimal in (13)], there must exist a direction of descent $\dot{Z} \in \mathbb{S}^{(d+1)n}$ from Z that is *not* captured in the low-rank factorization (16), *at least to first order* (since X is stationary). This could be because X is a saddle point of the nonconvex problem (16) (in which case there may exist a *second*-order direction of descent from X), or because the descent direction $\dot{Z}$ is toward a set of higher-rank matrices than the rank-r factorization used in (16) is able to capture. Part (b) of Theorem 5 provides an approach that enables us to address *both* of these potential obstacles simultaneously, by using a negative eigenvector of the certificate matrix $S(X)$ to construct a *second*-order direction of descent $\dot{X}_+$ from X_+, the lifting of X to the next (higher-rank) "step" of the Riemannian Staircase. Geometrically, this construction is based upon the (easily verified) fact that $S(X)$ is the Hessian of the Lagrangian $\nabla^2_X \mathcal{L}$ of the extrinsic (constrained) form of (16), and therefore, $\langle \dot{X}_+, \nabla^2_X \mathcal{L}\, \dot{X}_+ \rangle = \langle v, S(X) v \rangle < 0$, so that $\dot{X}_+$ is indeed a direction of second-order descent from the lifted stationary point X_+ [53], [55], [56].

In summary, Theorem 5 enables us to determine whether a first-order critical point X of (16) corresponds to a minimizer $Z = X^\top X$ of (13), and to *descend* from X if necessary, by computing the minimum eigenpair (λ, v) of the certificate matrix $S(X)$ defined in (34). In the original SE-Sync algorithm, the corresponding minimum-eigenvalue computation is performed by means of spectrally shifted Lanczos iterations [13], [57]; while this works well for the centralized SE-Sync method, adopting the Lanczos algorithm in the distributed setting would require an excessive degree of communication among the agents. Therefore, in the following section, we investigate alternative strategies for computing the minimum eigenpair that are more amenable to a distributed implementation. We also refer the reader to the technical report [1, Sec. 6] for additional discussions.

A. Distributed Minimum-Eigenvalue Computation

In this section, we describe an efficient distributed algorithm for computing the minimum eigenpair of the certificate matrix $S(X)$ required in Theorem 5. The simplest method suitable for distributed computation that we consider is the well-known power method [58, Sec. 8.2]. Each iteration computes the following matrix-vector product:[4]

$$x_{k+1} = Ax_k. \tag{37}$$

Note that with $A = S(X)$, the matrix-vector product in (37) can be computed using the same interagent communication pattern already employed in each iteration of the RBCD method developed in Section IV, and so is well-suited to a distributed implementation.

However, the power method's simplicity comes at the expense of a reduced convergence rate versus the Lanczos procedure. In particular, if $\gamma \in (0, 1)$ denotes the relative gap between the absolute values of the largest- and second-largest-magnitude eigenvalues of A, then the power method converges at the rate $\mathcal{O}(1/\gamma)$, while the Lanczos method converges as $\mathcal{O}(1/\sqrt{\gamma})$ (see [1, Sec. 6.1] for details). In our target application (certifying the optimality of a first order-critical point X), the minimum eigenvalue will *always* belong to a tight cluster ($\gamma \ll 1$) whenever X is a global minimizer,[5] so the power method's $\mathcal{O}(1/\gamma)$ rate translates to a substantial reduction in practical performance versus the Lanczos method's $\mathcal{O}(1/\sqrt{\gamma})$ rate.

To improve the convergence, we propose to adopt the recently developed *accelerated power method* [59] as our distributed eigenvalue algorithm of choice. In brief, this method modifies the standard power iteration (37) by adding a Polyak momentum term, producing the iteration:

$$x_{k+1} = Ax_k - \beta x_{k-1} \tag{38}$$

where $\beta \in \mathbb{R}_+$ is a *fixed* constant. We note that because β is constant, the iteration (38) has the same communication pattern as the standard power method (37), and so is well-suited to implementation in the distributed setting. Furthermore, it is shown in [59] that with a well-chosen parameter β (in particular, with $\beta \approx \lambda_2^2/4$, where λ_2 is the second dominant eigenvalue of A), the addition of momentum actually allows the accelerated power method to *match* the $\mathcal{O}(1/\sqrt{\gamma})$ dependence of the "gold-standard" (centralized) Lanczos procedure on the dominant eigengap.

Algorithm 6: Minimum Eigenpair (MinEig).

Input: Certificate matrix $S = S(X)$ from (34b).
Output: Minimum eigenpair $(\lambda_{\min}, v_{\min})$ of S.
1: Compute dominant (maximum-magnitude) eigenpair $(\lambda_{\text{dom}}, v_{\text{dom}})$ of S using power iteration (37).
2: **if** $\lambda_{\text{dom}} < 0$ **then**
3: **return** $(\lambda_{\text{dom}}, v_{\text{dom}})$
4: **end if**
5: Compute maximum eigenpair (θ, v) of $C \triangleq \lambda_{\text{dom}} I - S$ using accelerated power iteration (38).
6: **return** $(\lambda_{\text{dom}} - \theta, v)$

Combining the power and accelerated power methods with the spectral shifting strategy proposed in [57, Sec. III-C] produces our distributed minimum-eigenvalue method (see Algorithm 6). In brief, the main idea is to construct a spectrally shifted version C of the certificate matrix $S(X)$ such that, first, a maximum eigenvector v of C coincides with a *minimum* eigenvector of S, and, second, $C \succeq 0$, so that we can recover the maximum eigenpair (θ, v) of C using accelerated power iterations (38). Algorithm 6 accomplishes this by first applying the (basic) power method (37) to estimate the dominant eigenpair $(\lambda_{\text{dom}}, v_{\text{dom}})$ of S in line 6 (which does *not* require $S \succeq 0$), and then applying the accelerated power method to compute the maximum eigenpair (θ, v) of $C = \lambda_{\text{dom}} I - S \succeq 0$ in line 6. Note that while the minimum eigenvalue of $S(X)$ belongs to a tight cluster whenever X is optimal for (16) (necessitating our use of *accelerated* power iterations in line 6), the *dominant* eigenvalue of S is typically well-separated, and, therefore, can be computed to high precision using only a small number of power iterations in line 6.

Remark 9 (Communication requirements of Algorithm 6): The bulk of the work in Algorithm 6 lies in updating the eigenvector estimate via the matrix-vector products (37) and (38). In the distributed regime, these can be implemented by having each robot estimate the block of the eigenvector that corresponds to its own poses. The communication pattern of this process is determined by the sparsity structure of the underlying matrix, which for our application are the dual certificate S and its spectrally shifted version C. Fortunately, both S and C inherit the sparsity of the connection Laplacian Q. This means that at each iteration of (37) or (38), each robot only needs to communicate with its neighbors in the global pose graph. Therefore, Algorithm 6 provides an efficient way (in terms of *both* computation *and* communication) to compute a minimum eigenpair of S in the distributed setting.

[4]Note that while the power method iteration is commonly written in the normalized form $x_{k+1} = Ax_k/\|Ax_k\|$, normalization is only actually required to compute the Ritz value $\theta_k = x_k^\top Ax_k/\|x_k\|^2$ associated with x_k.

[5]This is an immediate consequence of the (extrinsic) first-order criticality condition for (16), which requires $S(X)X^\top = 0$, i.e., that *each row* of X be an eigenvector of $S(X)$ with eigenvalue 0 [57, Sec. III-C].

Algorithm 7: Descent From a Suboptimal Critical Point X_+ (EscapeSaddle).

Input:
- Lifted suboptimal critical point X_+ as defined in (35).
- Second-order descent direction $\dot{X}_+$ as defined in (36).

Output: Feasible point $X \in \mathcal{M}_{\text{PGO}}(r+1, n)$ satisfying $f(X) < f(X_+)$, $\|\operatorname{grad} f(X)\| > 0$.

1: Set initial stepsize: $\alpha = 1$.
2: Set initial trial point: $X \leftarrow \operatorname{Retr}_{X_+}(\alpha \dot{X}_+)$
3: **while** $f(X) \geq f(X_+)$ or $\|\operatorname{grad} f(X)\| = 0$ **do**
4: Halve steplength: $\alpha \leftarrow \alpha/2$.
5: Update trial point: $X \leftarrow \operatorname{Retr}_{X_+}(\alpha \dot{X}_+)$.
6: **end while**
7: **return** X.

B. Descent From Suboptimal Critical Points

In the event that $Z = X^\top X$ is *not* a minimizer of Problem 2 [as determined by $\lambda < 0$, where λ is the minimum eigenvalue of the certificate $S(X)$], we perform a distributed backtracking line-search along the descent direction $\dot{X}_+$ identified in Theorem 5(b). Algorithm 7 summarizes this procedure. We note that even though Algorithm 7 requires coordination among all of the agents (to evaluate the objective $f(X(\alpha))$ and gradient norm $\|\operatorname{grad} f(X(\alpha))\|$ each trial point $X(\alpha)$, and to distribute the trial stepsize α), it requires a sufficiently small number of (very lightweight) globally synchronized messages to remain tractable in the distributed setting.

VI. Distributed Initialization and Rounding

Various initialization techniques have been proposed for PGO [60]. In this work, we adopt the distributed *chordal initialization* used in [6]. From an initial estimate $T \in \mathrm{SE}(d)^n$, we obtain an equivalent initial solution on $\mathcal{M}_{\text{PGO}}(r, n)$ via $X = Y_{\text{rand}} T$, where $Y_{\text{rand}} \in \mathrm{St}(d, r)$ is a random element of the Stiefel manifold. The reader is referred to [1, Sec. 7.1] for more details.

After solving the SDP relaxation, we need to "round" the low-rank factor $X^\star \in \mathcal{M}_{\text{PGO}}(r, n)$ returned by the Riemannian Staircase to a feasible solution to the original PGO problem (see line 2 in Algorithm 2). In this section, we describe a distributed rounding procedure that incurs minimal computation and communication costs, and furthermore is guaranteed to return a global minimizer to the original PGO (see Problem 1) provided that the SDP relaxation is *exact*.

Given the output $X^\star$ from the Riemannian Staircase, consider its individual components that correspond to the "lifted" rotation and translation variables,

$$X^\star = \begin{bmatrix} Y_1^\star & p_1^\star & \dots & Y_n^\star & p_n^\star \end{bmatrix} \in (\mathrm{St}(d, r) \times \mathbb{R}^r)^n. \quad (39)$$

In Theorem 2, we have shown that if the SDP relaxation is exact, then the first block-row of the corresponding SDP solution, which can be written as $T^\star \triangleq (Y_1^\star)^\top X^\star$, gives a global minimizer to PGO (see Problem 1). Looking at the rotation and translation of each pose in $T^\star$ separately,

$$R_i^\star = (Y_1^\star)^\top Y_i^\star, \; t_i^\star = (Y_1^\star)^\top p_i^\star. \quad (40)$$

Equation (40), thus, recovers globally optimal rotation and translation estimates. If the SDP relaxation is not exact, the $R_i^\star$ as computed in (40) may not be a valid rotation. To ensure feasibility, we additionally project it to $\mathrm{SO}(d)$,

$$R_i = \operatorname{Proj}_{\mathrm{SO}(d)}(Y_1^{\star\top} Y_i^\star). \quad (41)$$

In (41), the projection can be carried out by computing the SVD.

Remark 10 (Communication requirements of distributed initialization and rounding): The distributed chordal initialization [6] has a similar communication pattern as distributed local search, where at each iteration robots exchange messages corresponding to their public poses. On the other hand, distributed rounding incurs minimal communication, since agents only need to relay the small r-by-d matrix $Y_1^\star$ over the network.

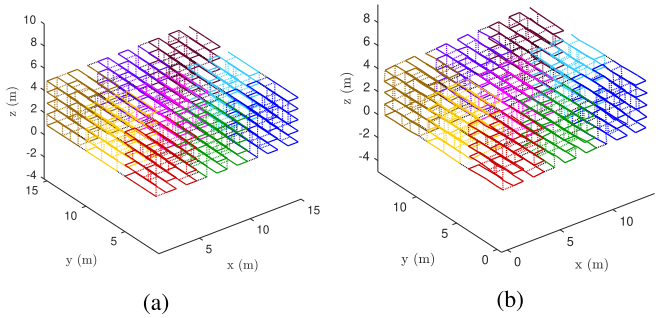
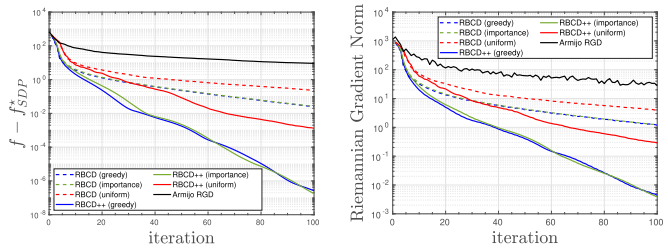


Fig. 4. Example simulation consisting of nine robots (trajectories shown in different colors) where each robot has 125 poses. Loop closures are drawn as dotted lines. (a) Ground truth. (b) Certified global minimizer returned by DC2-PGO (see Algorithm 2). (a) Ground truth. (b) Certified solution from Algorithm 2.

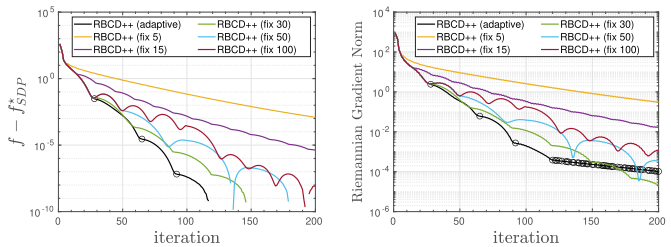
VII. Experiments

We perform extensive evaluations of the proposed DC2-PGO algorithm on both simulations and benchmark CSLAM datasets. Our simulation consists of multiple robots moving next to each other in a 3-D grid with lawn mower trajectories. With a given probability (default 0.3), loop closures are added to connect neighboring poses. For all relative measurements, we simulate isotropic Langevin rotation noise according to (6) with mode I_d and concentration parameter κ. To make the process of setting κ more intuitive, we first set a desired standard deviation σ_R for the *rotation angle* of the rotational noise, and then use the asymptotic approximation shown in SE-Sync [13, App. A] to compute the corresponding concentration parameter κ. We also simulate Gaussian translation noise according to (7) with zero mean and standard deviation σ_t. The default noise parameters are $\sigma_R = 3^\circ, \sigma_t = 0.05$ m. See Fig. 4 for an example simulation together with the certified global minimizer found by DC2-PGO (see Algorithm 2). All implementations are done in MATLAB. All experiments are carried out on a laptop with an Intel i7 CPU and 8 GB RAM.



(a) (b)

Fig. 5. Convergence rates of RBCD and RBCD++ with uniform, importance, or greedy selection rules, on the nine robot simulation shown in Fig. 4. (a) Evolution of optimality gap averaged averaged over ten random runs. (b) Evolution of Riemannian gradient norm averaged over ten random runs. (a) Average optimality gap. (b) Average gradient norm.



(a) (b)

Fig. 6. Adaptive restart versus fixed restart for RBCD++ (greedy selection) on a random instance of our simulation. For fixed restart, we use restart frequency ranging from every 5 to every 100 iterations. For adaptive restart (black curves), we also highlight iterations where restart is triggered with circle markers. (a) Evolution of optimality gaps. (b) Evolution of Riemannian gradient norm. (a) Optimality Gap. (b) Gradient norm.

When evaluating our approach and baseline methods, we use the following performance metrics. First, we compute the optimality gap $f - f^\star_{\text{SDP}}$, where $f^\star_{\text{SDP}}$ is the optimal value of the centralized SDP relaxation computed using SE-Sync [13]. In the (typical) case that the SDP relaxation is exact, the rounded PGO estimates returned by both our approach and SE-Sync will achieve a zero optimality gap (to within numerical tolerances). Additionally, when evaluating convergence rates of local search methods, we compute the evolution of the Riemannian gradient norm, which quantifies how fast the iterates are converging to a first-order critical point. Lastly, additional results that evaluate the estimation root-mean-square errors may be found in the technical report [1, Sec. 8].

A. *Evaluations of Distributed Local Search*

We first evaluate the performance of the proposed RBCD and RBCD++ algorithms when solving the rank-restricted relaxations (see Problem 4). Recall that this serves as the central local search step in our overall Riemannian Staircase algorithm. By default, we set the relaxation rank to $r = 5$, and enable parallel execution as described in Section IV-D.

Fig. 5 shows the performance on our nine robot scenario shown in Fig. 4. We report the performance of our proposed methods using all three block selection rules proposed in Section IV-A: uniform sampling, importance sampling, and greedy selection. For reference, we also compare our performance against the RGD algorithm with Armijo's backtracking line search implemented in Manopt [61]. As the results demonstrate, both RBCD and RBCD++ dominate the baseline RGD algorithm in terms of the convergence speed and solution quality. As expected, importance sampling and greedy block selection also lead to faster convergence compared to uniform sampling. Furthermore, RBCD++ shows significant empirical acceleration and is able to converge to the global minimum with high precision using only 100 iterations. Note that in this experiment, we choose to report convergence speed with respect to iteration number, because it is directly linked to the number of communication rounds required during distributed optimization. For completeness, we also note that the average runtime of BLOCKUPDATE (based on a modified version of the trust-region solver in Manopt [61]) is 0.023 s.

In Fig. 5, we report the performance of RBCD++ with the default adaptive restart scheme (see Algorithm 5). As we have discussed in Remark 6, a less expensive and, hence, more practical restart scheme is simply to reset Nesterov's acceleration after a fixed number of iterations; this scheme is typically referred to as *fixed restart* in the literature. In Fig. 6, we compare adaptive restart with fixed restart on a random instance of our simulation. For fixed restart, we use different restart frequency ranging from every 5 to every 100 iterations. We observe that with a short restart period (e.g., 5), convergence of RBCD++ is significantly slowed down. This result is expected, as frequent restarting essentially removes the effect of acceleration from the iterations of RBCD++. In the extreme case of restarting at every iteration, the algorithm essentially reduces to RBCD. On the other hand, the long restart period (e.g., 100) also has a negative impact, and we observe that the overall convergence displays undesirable oscillations. Finally, we find that a suitably chosen restart period (e.g., 30) demonstrates a superior convergence rate that is similar to the adaptive restart scheme.

On the same nine robot scenario, we also report the convergence of RBCD and RBCD++ (using greedy selection) under increasing measurement noise, shown in Fig. 7. As expected, as rotation noise increases, the convergence rates of both RBCD and RBCD++ are negatively impacted. On the other hand, we observe that increasing translation noise actually leads to better convergence behavior, as shown in Fig. 7(c) and (d). To explain these observations, we conjecture that increasing rotation noise and decreasing translation noise magnify the *ill conditioning* of the optimization problem. Qualitatively similar results were also reported in [13] (decreasing translational noise was observed to *increase* SE-Sync's wall-clock time).

Lastly, we evaluate the scalability of RBCD and RBCD++ (both with greedy block selection) as the number of robots increases from 4 to 49 in the simulation. As each robot has 125 poses, the maximum size of the global PGO problem is 6125. Fig. 8 reports the convergence speeds measured in Riemannian gradient norm. Both RBCD and RBCD++ are reasonably fast for small number of robots. Nevertheless, the nonaccelerated RBCD

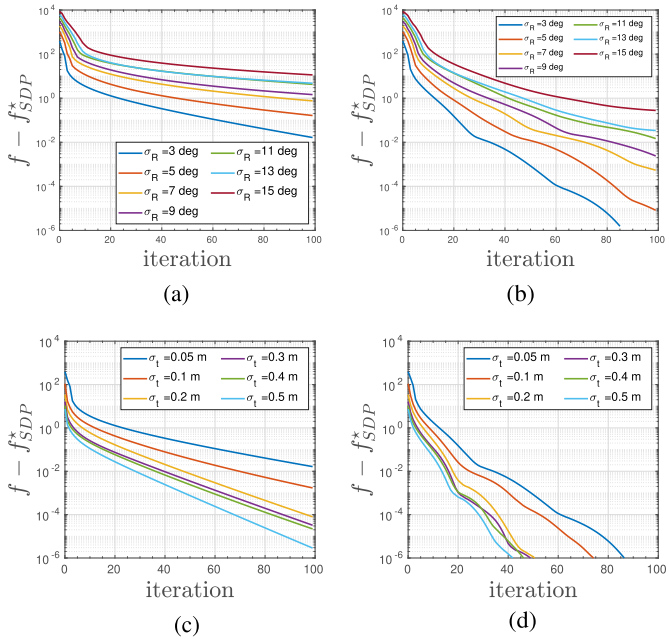


Fig. 7. Convergence of RBCD and RBCD++ with greedy selection under varying rotation and translation measurement noise. (a) and (b) Convergence of RBCD and RBCD++ under increasing rotation noise and fixed translation noise of $\sigma_t = 0.05$ m. (c) and (d) Convergence of RBCD and RBCD++ under increasing translation noise and fixed rotation noise of $\sigma_t = 3^\circ$. All results are averaged across ten random runs.

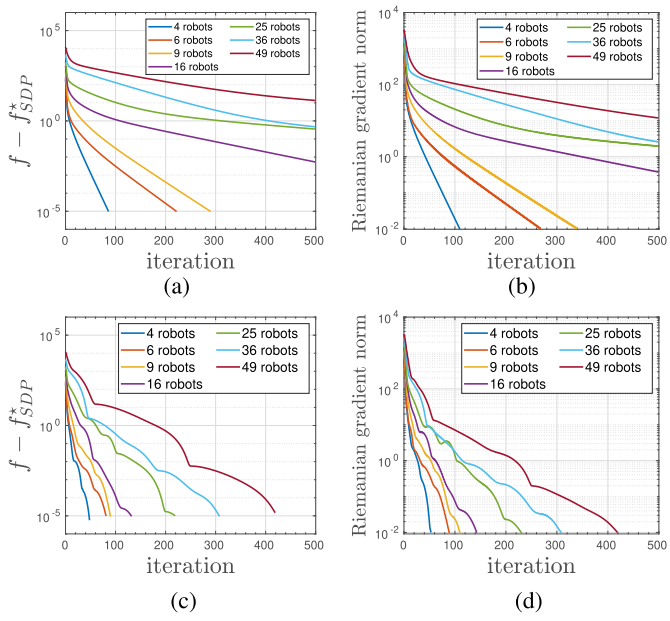


Fig. 8. Scalability of RBCD and RBCD++ with greedy selection as the number of robots increases. Each robot has 125 poses. Convergence speed is measured in terms of the Riemannian gradient norm. (a) RBCD optimality gap. (b) RBCD gradient norm. (c) RBCD++ optimality gap. (d) RBCD++ gradient norm.

algorithm begins to show slow convergence as the number of robots exceeds 16. In comparison, our accelerated RBCD++ algorithm shows superior empirical convergence speed, even in the case of 49 robots. We note that in this case although RBCD++ uses 400 iterations to achieve a Riemannian gradient norm of 10^{-2}, the actual optimality gap [see Fig. 8(c)] decreases much more rapidly to 10^{-5}, which indicates that our solution is very close to the global minimum.

B. Evaluations of Distributed Verification

In this section, we evaluate our proposed distributed verification method. Recall from Section V that the bulk of work happens when using accelerated power iteration to compute the dominant eigenpair of the spectrally shifted dual certificate matrix $C \triangleq \lambda_{\text{dom}} I - S(X)$. We, thus, examine the efficiency of this process, and compare the performance of accelerated power iteration against standard power method and the centralized Lanczos procedure. We note that since C and $S(X)$ share the same set of eigenvectors, in our experiment, we still report results based on the estimated eigenvalues of $S(X)$.

From [59], the accelerated power iteration (38) achieves the theoretical optimal rate when the momentum term satisfies $\beta \approx \lambda_2^2/4$, where λ_2 is the second dominant eigenvalue of C. Since we know that λ_{dom} belongs to a tight cluster whenever X is globally optimal, we expect that typically $\lambda_2 \approx \lambda_{\text{dom}}$. Using this insight, in our experiment, we first estimate λ_2 by multiplying λ_{dom} with a factor $\gamma < 1$ that is close to one, i.e., $\widehat{\lambda}_2 = \gamma \lambda_{\text{dom}}$. Subsequently, we use this estimated value to set the momentum term,

$$\beta = \widehat{\lambda}_2^2/4 = \gamma^2 \lambda_{\text{dom}}^2/4. \tag{42}$$

We design two test cases using the `Killian court` dataset. In the first case, we verify the global minimizer computed by SE-Sync [13]. By Theorem 5, the dual certificate matrix $S(X)$ must be positive semidefinite. Furthermore, since $S(X)$ always has a nontrivial nullspace spanned by the rows of the corresponding primal solution, we expect the minimum eigenvalue of $S(X)$ to be zero in this case. Indeed, when computing this using the `eigs` function in MATLAB, the final value [denoted as $\lambda^\star$ in Fig. 9(a)] is close to zero to machine precision. Fig. 9(a) shows how fast each method converges to $\lambda^\star$, where we use an initial eigenvector estimate obtained by (slightly) randomly perturbing a row of the global minimizer [57]. Fig. 9(b) shows the corresponding Ritz residual for the estimated eigenvector v, defined as follows:

$$\text{ResidualNorm}(v) = \|S(X)v - (v^\top S(X) v)v\|_2 \tag{43}$$

where v is normalized. The results suggest that, with a suitable choice of γ, the accelerated power iteration is significantly faster than the standard power method. Furthermore, in this case, convergence speed is close to the Lanczos procedure.

In the second case, we verify a suboptimal first-order critical point obtained by running RBCD++ with $r = d$ using random initialization. We verify that the minimum eigenvalue of $S(X)$ is negative (≈ -2.97), which is consistent with the prediction of Theorem 5. In this case, we observe that using a *random* initial eigenvector estimate leads to better convergence compared to obtaining the initial estimate from a perturbed row of the primal solution. Intuitively, using the perturbed initial guess would

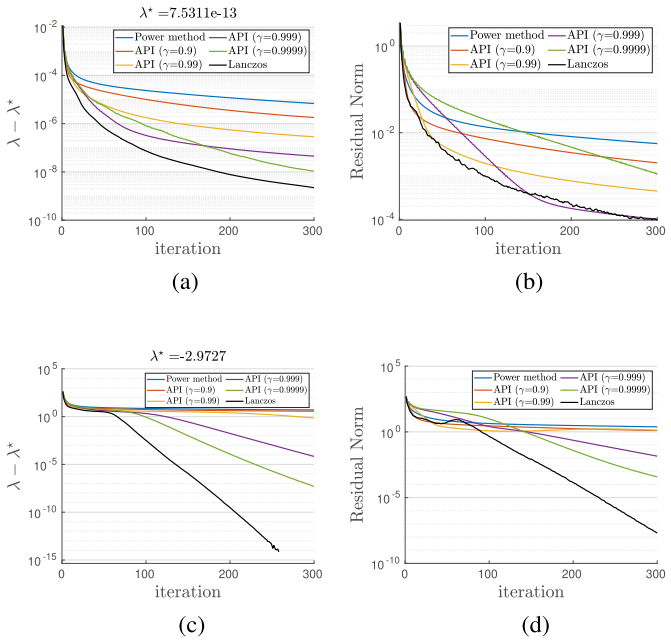


Fig. 9. Performance of accelerated power iteration (API) on the dataset, using different values of γ to set the momentum term according to (42). (a) and (b) Verification of a global minimizer computed by SE-Sync. (c) and (d) Verification of a suboptimal first-order critical point. In both cases, the minimum eigenvalue of the dual certificate matrix (denoted as $\lambda^\star$) is computed using the function in MATLAB.

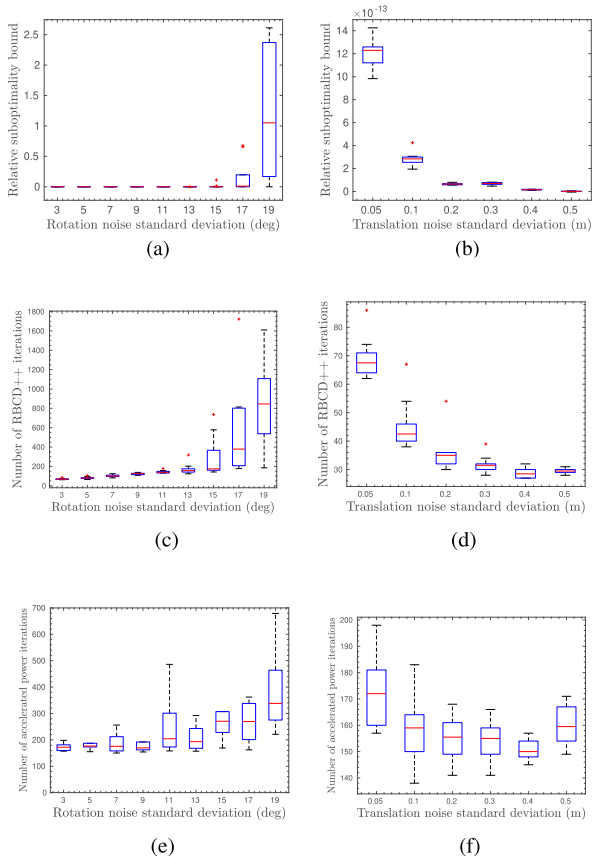


Fig. 10. Evaluation of the proposed DC2-PGO (see Algorithm 2) under increasing measurement noise. At each noise level, we simulate ten random realizations of the nine-robot scenario shown in Fig. 4. Top row shows boxplots of relative suboptimality bound $(f(T) - f^\star_{\text{SDP}})/f^\star_{\text{SDP}}$. Middle and bottom rows show boxplots of total number of iterations used by RBCD++ and accelerated power method. The left column shows results under increasing rotation noise $\sigma_R \in [3, 19]^\circ$ and fixed translation noise $\sigma_t = 0.05$ m. The right column shows results under increasing translation noise $\sigma_t \in [0.05, 0.5]$ m and fixed rotation noise $\sigma_R = 3$ deg. (a) Relative suboptimality bound under increasing rotation noise. (b) Relative suboptimality bound under increasing translation noise. (c) Total number of RBCD++ iterations under increasing rotation noise. (d) Total number of RBCD++ iterations under increasing translation noise. (e) Total number of accelerated power iterations under increasing rotation noise. (f) Total number of accelerated power iterations under increasing translation noise.

cause the iterates of power method to be "trapped" for longer periods of time near the zero eigenspace spanned by the rows of X. Fig. 9(c) and (d) shows results generated with random initial eigenvector estimate. Note that there is a bigger performance gap between accelerated power iteration and the Lanczos algorithm. However, in reality, full convergence is also not necessary in this case. Indeed, from Theorem 5, we need only identify *some* direction that satisfies $v^\top S(X)v < 0$ in order to escape the current suboptimal solution.

C. Evaluations of Complete Algorithm (Algorithm 2)

So far, we have separately evaluated the proposed local search and verification techniques. In this section, we evaluate the performance of the complete DC2-PGO algorithm (see Algorithm 2) that uses distributed Riemannian Staircase (see Algorithm 1) to solve the SDP relaxation of PGO. By default, at each level of the Staircase, we use RBCD++ with greedy selection to solve the rank-restricted relaxation until the Riemannian gradient norm reaches 10^{-1}. Then, we use the accelerated power method to verify the obtained solution. To set the momentum term, we employ the same method introduced in the last section with $\gamma = 0.999$ in (42). The accelerated power iteration is deemed converged when the eigenvector residual defined in (43) reaches 10^{-2}.

We first examine the exactness of the SDP relaxation in the nine-robot scenario shown in Fig. 4 under increasing measurement noise. Recall that DC2-PGO returns both a rounded feasible solution $T \in \text{SE}(d)^n$ as well as the optimal value of the SDP relaxation $f^\star_{\text{SDP}}$. To evaluate exactness, we record the upper bound on the relative suboptimality of T, defined as $(f(T) - f^\star_{\text{SDP}})/f^\star_{\text{SDP}}$. We note that a zero suboptimality bound means that the SDP relaxation is exact and the solution T is a global minimizer. As shown in the first row of Fig. 10, DC2-PGO is capable of finding global minimizers for all translation noise considered in our experiments, and for rotation noise up to 11 °, which is still much larger than noise magnitude typically encountered in SLAM. The second and third rows of Fig. 10 show the total number of iterations used by RBCD++ and accelerated power method, across all levels of the staircase. Interestingly, we observe that changing measurement noise has a greater impact for distributed local search compared to distributed verification.

Lastly, we evaluate DC2-PGO on benchmark datasets. Visualizations of these datasets are shown in the technical report [1, Fig. 11]. In Table I, we compare the performance of DC2-PGO against the centralized certifiable SE-Sync algorithm [13], as

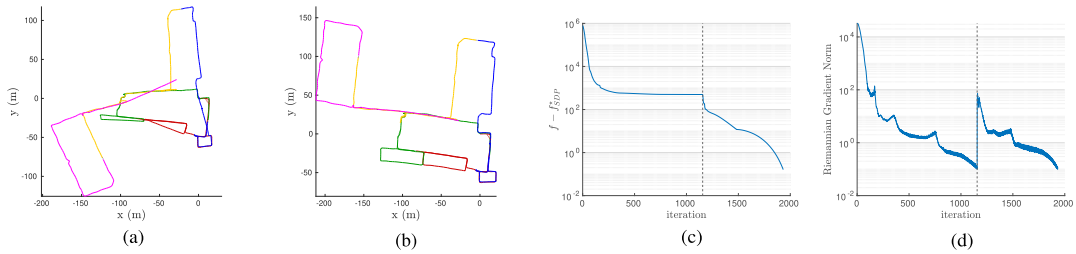


Fig. 11. DC2-PGO (see Algorithm 2) returns the global minimizer of the Killian court dataset even from random initialization. (a) From random initialization, RBCD++ converges to a suboptimal critical point at rank $r = 3$. (b) Via distributed verification and saddle point escaping, our algorithm is able to escape the suboptimal solution and converges to the global minimizer at rank $r = 4$. (c) Evolution of optimality gap across $r = 3$ and $r = 4$. (d) Evolution of Riemannian gradient norm across $r = 3$ and $r = 4$. The vertical dashed line indicates the transition from $r = 3$ to $r = 4$.

TABLE I
EVALUATION ON BENCHMARK PGO DATASETS

Dataset	# Vertices	# Edges	Objective				Local Search Iterations	
			Init.	SE-Sync [13]	DGS [6]	DC2-PGO	DGS [6]	DC2-PGO
Killian Court (2D)	808	827	229.0	61.15	63.52	**61.22**	27105	**189**
CSAIL (2D)	1045	1171	31.50	31.47	31.49	**31.47**	**10**	197
Intel Research Lab (2D)	1228	1483	396.6	393.7	428.89	**393.7**	**10**	187
Manhattan (2D)	3500	5453	369.0	193.9	242.05	**194.0**	1585	**785**
KITTI 00 (2D)	4541	4676	1194	125.7	269.87	**125.7**	**1485**	2750
City10000 (2D)	10000	20687	5395	638.6	2975.2	**638.7**	2465	**1646**
Parking Garage (3D)	1661	6275	1.64	1.263	1.33	**1.311**	**25**	47
Sphere (3D)	2500	4949	1892	1687	1689	**1687**	300	**53**
Torus (3D)	5000	9048	24617	24227	24246	**24227**	100	**88**
Cubicle (3D)	5750	16869	786.0	717.1	726.69	**717.1**	**45**	556
Rim (3D)	10195	29743	8177	5461	5960.4	**5461**	**515**	1563

Each dataset simulates a CSLAM scenario with five robots. On each dataset, we report the objective value achieved by initialization, centralized SE-Sync [13], the DGS With SOR parameter 1.0 as recommended in [6], and the proposed DC2-PGO algorithm. For the latter two distributed algorithms, we also report the total number of local search iterations. On each dataset, we highlight the distributed algorithm that (i) achieves lower objective and (ii) uses less local search iterations. On all datasets, our approach is able to verify its solution as the global minimizer. We note that the numerical difference with SE-Sync on some datasets is due to the looser convergence condition during distributed local search.

well as the state-of-the-art distributed Gauss–Seidel (DGS) algorithm by Choudhary *et al.* [6]. For DGS, we set the SOR parameter to 1.0 as recommended by the authors, and for which we also observe stable performance in general. On all datasets, DC2-PGO is able to verify its solution as the global minimizer. We note that on some datasets, the final objective value is slightly higher than SE-Sync. This is due to the looser convergence condition used in our distributed local search: for RBCD++ we set the gradient norm threshold to 10^{-1}, while for SE-Sync we set the threshold to 10^{-6} in order to obtain a high-precision reference solution.[6] On the other hand, DC2-PGO is clearly more advantageous compared to DGS, as it returns a global minimum, often with fewer iterations. The performance of DGS is more sensitive to the quality of initialization, as manifested on the `Killian Court` and `City10000` datasets. In addition, while our local search methods are guaranteed to reduce the objective value at each iteration, DGS does *not* have this guarantee as it is operating on a linearized approximation of the PGO problem.

[6]In general it is not reasonable to expect RBCD or RBCD++ to produce solutions that are as precise as those achievable by SE-Sync in tractable time, since the former are *first-order* methods, while the latter is *second-order*.

To further demonstrate the uniqueness of our algorithm as a *global* solver, we show that it is able to converge to the global minimum even from random initialization. This is illustrated using the `Killian court` dataset in Fig. 11. Due to the random initialization, the first round of distributed local search at rank $r = 3$ converges to a suboptimal critical point. This can be seen in Fig. 11(c), where the optimality gap at $r = 3$ (first 1156 iterations) converges to a nonzero value. From distributed verification, our algorithm detects the solution as a saddle point and is able to escape and converges to the correct global minimizer at rank $r = 4$. This can be clearly seen in Fig. 11(d), where escaping successfully moves the iterate to a position with large gradient norm, from where local search can successfully descend to the global minimizer.

VIII. CONCLUSION

In this work, we proposed the first *certifiably correct* algorithm for *distributed* PGO. Our method was based upon a sparse semidefinite relaxation of the PGO problem that we proved enjoyed the same exactness guarantees as current state-of-the-art centralized methods [13]: namely, that its minimizers were *low-rank* and provided *globally optimal solutions* of the original PGO problem under moderate noise. To solve large-scale instances of

this relaxation in the distributed setting, we leveraged the existence of low-rank solutions to propose a distributed Riemannian Staircase framework, employing RBCD as the core distributed optimization method. We proved that RBCD enjoyed a *global* sublinear convergence rate under standard (mild) conditions, and could be significantly accelerated using Nestorov's scheme. We also developed the first distributed solution verification and saddle escape algorithms to *certify* the optimality of critical points recovered via RBCD, and to descend from suboptimal critical points if necessary. Finally, we provided extensive numerical evaluations, demonstrating that the proposed approach correctly recovers globally optimal solutions under moderate noise, and outperformed alternative distributed methods in terms of estimation quality and convergence speed.

In the future, we plan to study approaches to further improve iteration complexity on hard problem instances. Another interesting direction is the design of certifiably correct distributed algorithm that can handle *outlier* measurements in CSLAM and CNL. While robustness has gained increasing attention in multirobot SLAM [31], [62], recent *certifiably robust* approaches such as [63] are still restricted to centralized setting due to high computational requirements.

Lastly, similar to other distributed algorithms, the performance of DC2-PGO is expected to degrade as network conditions deteriorate (e.g., as network topology becomes sparser or delay increases). To this end, our recent work [64] studied convergence of distributed PGO under communication delay. In general, however, designing distributed SLAM systems that better handle real-world communication challenges remains an important direction for future research.

ACKNOWLEDGMENT

The authors would like to thank Prof. L. Carlone for fruitful discussions that led to this work.

REFERENCES

[1] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, "Distributed certifiably correct pose-graph optimization," 2019. [Online]. Available: https://arxiv.org/abs/1911.03721.

[2] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *J. Field Robot.*, vol. 36, no. 4, pp. 763–781, 2019.

[3] I. Deutsch, M. Liu, and R. Siegwart, "A framework for multi-robot pose graph SLAM," in *Proc. IEEE Int. Conf. Real-time Comput. Robot.*, 2016, pp. 567–572.

[4] J. G. Morrison, D. Gálvez-López, and G. Sibley, "MOARSLAM: Multiple operator augmented RSLAM," in *Distributed Autonomous Robotic Systems*. Berlin, Germany: Springer, 2016, pp. 119–132.

[5] B. Kim *et al.*, "Multiple relative pose graphs for robust cooperative mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 3185–3192.

[6] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "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.

[7] T. Fan and T. Murphey, "Majorization minimization methods to distributed pose graph optimization with convergence guarantees," *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 5058–5065, doi: 10.1109/IROS45743.2020.9341063.

[8] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5220–5227.

[9] 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.

[10] T. Fan and T. D. Murphey, "Generalized proximal methods for pose graph optimization," in *Proc. Int. Symp. Robot. Res.*, 2019.

[11] 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.

[12] A. S. Bandeira, "A note on probably certifiably correct algorithms," *Comptes Rendus Mathematique*, vol. 354, no. 3, pp. 329–333, 2016.

[13] 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.

[14] J. Briales and J. Gonzalez-Jimenez, "Cartan-sync: Fast and global SE(d)-synchronization," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2127–2134, Oct. 2017.

[15] N. Boumal, "A Riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints," 2015, *arXiv:1506.00575*.

[16] N. Boumal, "An Introduction to Optimization on Smooth Manifolds," Aug. 2020. [Online]. Available: http://www.nicolasboumal.net/book

[17] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[18] S. Burer and R. D. C. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Math. Prog.*, vol. 95, pp. 329–357, 2003.

[19] P.-A. Absil, C. G. Baker, and K. A. Gallivan, "Trust-region methods on Riemannian manifolds," *Found. Comput. Math.*, vol. 7, no. 3, pp. 303–330, 2007.

[20] K. Khosoussi, S. Huang, and G. Dissanayake, "A sparse separable SLAM back-end," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1536–1549, Dec. 2016.

[21] A. Singer, "Angular synchronization by eigenvectors and semidefinite programming," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 1, pp. 20–36, 2011.

[22] A. S. Bandeira, N. Boumal, and A. Singer, "Tightness of the maximum likelihood semidefinite relaxation for angular synchronization," *Math. Program.*, vol. 163, no. 1, pp. 145–167, May 2017.

[23] A. Eriksson, C. Olsson, F. Kahl, and T. Chin, "Rotation averaging with the chordal distance: Global minimizers and strong duality," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 256–268, Jan. 2021.

[24] F. Dellaert, D. Rosen, J. Wu, R. Mahony, and L. Carlone, "Shonan rotation averaging: Global optimality by surfing $SO(p)^n$," in *Proc. Eur. Conf. Comput. Vis.*, 2020.

[25] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *Int. J. Comput. Vis.*, vol. 103, no. 3, pp. 267–305, 2013.

[26] R. Tron, *Distributed Optimization on Manifolds for Consensus Algorithms and Camera Network Localization*. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2012.

[27] 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.

[28] R. Tron, J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "A distributed optimization framework for localization and formation control: Applications to vision-based measurements," *IEEE Control Syst. Mag.*, vol. 36, no. 4, pp. 22–44, Aug. 2016.

[29] 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.

[30] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2466–2473.

[31] 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. (RA-L)*, vol. 5, no. 2, pp. 1656–1663, 2020.

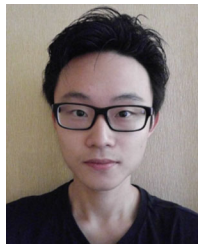
[32] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, vol. 23. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[33] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, 2012.

[34] B. O'Donoghue and E. Candes, "Adaptive restart for accelerated gradient schemes," *Found. Comput. Math.*, vol. 15, no. 3, pp. 715–732, 2012.

[35] S. J. Wright, "Coordinate descent algorithms," *Math. Prog.*., vol. 151, no. 1, pp. 3–34, Jun. 2015.

[36] J. Nutini, I. Laradji, and M. Schmidt, "Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence," 2017, *arXiv:1712.08859*.
[37] J. Mareček, P. Richtárik, and M. Takáč, "Distributed block coordinate descent for minimizing partially separable functions," in *Numerical Analysis and Optimization*. Berlin, Germany: Springer, 2015, pp. 261–288.
[38] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2037–2060, 2013.
[39] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, 2012.
[40] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 4, pp. 3841–3846.
[41] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 196–207, Apr. 2005.
[42] R. Hartley, K. Aftab, and J. Trumpf, "L1 rotation averaging using the Weiszfeld algorithm," in *Proc. IEEE CVPR*, 2011, pp. 3041–3048.
[43] P.-W. Wang, W.-C. Chang, and J. Z. Kolter, "The mixing method: Low-rank coordinate descent for semidefinite programming with diagonal constraints," 2017, *arXiv:1706.00476*.
[44] M. A. Erdogdu, A. Ozdaglar, P. A. Parrilo, and N. D. Vanli, "On the convergence of block-coordinate maximization for burer-monteiro method," *ICML Workshop*, Modern Trends in Nonconvex Optimization for Machine Learning, in 2018.
[45] Y. Tian, K. Khosoussi, and J. P. How, "Block-coordinate minimization for large SDPs with block-diagonal constraints," 2019, *arXiv:1903.00597*.
[46] M. Giamou, K. Khosoussi, and J. P. How, "Talk resource-efficiently to me: Optimal communication planning for distributed loop closure detection," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3841–3848.
[47] Y. Tian, K. Khosoussi, and J. P. How, "A resource-aware approach to collaborative loop closure detection with provable performance guarantees," *Int. J. Robot. Res.*, doi: 10.1177/0278364920948594.
[48] A. Javanmard, A. Montanari, and F. Ricci-Tersenghi, "Phase transitions in semidefinite relaxations," *Proc. Nat. Acad. Sci.*, vol. 113, no. 16, pp. E2218–E2223, 2016.
[49] N. A. Lynch, *Distributed Algorithms*. Amsterdam, The Netherlands: Elsevier, 1996.
[50] N. Boumal, P.-A. Absil, and C. Cartis, "Global rates of convergence for nonconvex optimization on manifolds," *IMA J. Numer. Anal.*, vol. 39, no. 1, pp. 1–33, Feb. 2018.
[51] L. Barenboim and M. Elkin, "Distributed $(\delta + 1)$-coloring in linear (in δ) time," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 111–120.
[52] J. Schneider and R. Wattenhofer, "A new technique for distributed symmetry breaking," in *Proc. 29th ACM SIGACT-SIGOPS Symp. Princ. Distributed Comput.*, 2010, pp. 257–266.
[53] D. M. Rosen, "Scalable low-rank semidefinite programming for certifiably correct machine perception," in *Proc. Int. Workshop Algorithmic Found. Robot.*, Jun. 2020.
[54] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, no. 1, pp. 49–95, Mar. 1996.
[55] N. Boumal, V. Voroninski, and A. S. Bandeira, "The non-convex Burer-Monteiro approach works on smooth semidefinite programs," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2765–2773.
[56] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre, "Low-rank optimization on the cone of positive semidefinite matrices," *SIAM J. Optim.*, vol. 20, no. 5, pp. 2327–2351, 2010.
[57] D. Rosen and L. Carlone, "Computational enhancements for certifiably correct SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2017, (Workshop on "Introspective Methods for Reliable Autonomy")*.
[58] G. Golub and C. V. Loan, *Matrix Computations*, *3rd ed.*Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.
[59] C. de Sa, B. He, I. Mitliagkas, C. Ré, and P. Xu, "Accelerated stochastic power iteration," in *Proc.* Mach. Learn. Res., 2018, vol. 84, pp. 58–67.
[60] 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. Autom.*, May 2015, pp. 4597–4604.
[61] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a MATLAB toolbox for optimization on manifolds," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1455–1459, 2014. [Online]. Available: http://www.manopt.org
[62] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 1–8.
[63] P. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling perceptual aliasing in SLAM via discrete-continuous graphical models," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1232–1239, Apr. 2019.
[64] 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.



Yulun Tian received the B.A. degree in computer science from UC Berkeley, Berkeley, CA, USA, in 2017, and the S.M. degree in aeronautics and astronautics in 2019 from the Massachusetts Institute of Technology, Cambridge, MA, USA, where he is currently working toward the Ph.D. degree in aeronautics and astronautics.

His current research interests include geometric estimation and optimization for distributed multiagent systems.

Kasra Khosoussi received the B.Sc. degree in computer engineering from the Department of Electrical and Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran, in 2011, and the Ph.D. degree in robotics from the Centre for Autonomous Systems (CAS), University of Technology Sydney, Ultimo, NSW, Australia, in 2017. He was a Visiting Ph.D. Student with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA (2015–2016).

He is currently a Research Scientist with the Department of Aeronautics and Astronautics, and the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA, USA, where he previously was a Postdoctoral Associate (2017–2018). His research is primarily focused on developing robust and reliable algorithms with provable performance guarantees for single- and multirobot perception. His work was a Best Paper Award finalist in multirobot systems at the 2018 IEEE International Conference on Robotics and Automation (ICRA).

David M. Rosen received the B.S. degree in mathematics from the California Institute of Technology, Pasadena, CA, USA, in 2008, the M.A. degree in mathematics from the University of Texas at Austin, Austin, TX, USA, in 2010, and the Sc.D. degree in computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2016.

He is currently a Postdoctoral Associate with the Department of Aeronautics and Astronautics and the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology. His research addresses the design of practical provably robust methods for machine perception and learning, using a combination of tools from optimization, geometry and topology, abstract algebra, and probability and statistics.

He was a recipient of the Best Paper Award at WAFR (2016), an RSS Pioneer Award (2019), and an RSS Best Student Paper Award (2020).

Jonathan P. How (Fellow, IEEE) received the B.A.Sc. degree in engineering science from the University of Toronto, Toronto, ON, Canada, in 1987, and the S.M. and Ph.D. degrees in aeronautics and astronautics from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1990 and 1993, respectively.

Prior to joining MIT in 2000, he was an Assistant Professor with Stanford University. He is currently the Richard C. Maclaurin Professor of Aeronautics and Astronautics with MIT.

Dr. How was the recipient of the IEEE CSS Distinguished Member Award (2020), AIAA Intelligent Systems Award (2020), IROS Best Paper Award on Cognitive Robotics (2019), and the AIAA Best Paper in Conference Awards (2011, 2012, 2013). He was the Editor-in-chief of *IEEE Control Systems Magazine* (2015–2019), is a Fellow of AIAA, and was elected to the National Academy of Engineering in 2021.