Spectral Sparsification for Communication-Efficient Collaborative Rotation and Translation Estimation

Yulun Tian †, Member, IEEE, and Jonathan P. How ‡, Fellow, IEEE

Abstract—We propose fast and communication-efficient optimization algorithms for multirobot rotation averaging and translation estimation problems that arise from collaborative simultaneous localization and mapping (SLAM), structure-from-motion (SfM), and camera network localization applications. Our methods are based on theoretical relations between the Hessians of the underlying Riemanian optimization problems and the Laplacians of suitably weighted graphs. We leverage these results to design a collaborative solver in which robots coordinate with a central server to perform approximate second-order optimization, by solving a Laplacian system at each iteration. Crucially, our algorithms permit robots to employ spectral sparsification to sparsify intermediate dense matrices before communication, and hence provide a mechanism to tradeoff accuracy with communication efficiency with provable guarantees. We perform rigorous theoretical analysis of our methods and prove that they enjoy (local) linear rate of convergence. Furthermore, we show that our methods can be combined with graduated nonconvexity to achieve outlier-robust estimation. Extensive experiments on real-world SLAM and SfM scenarios demonstrate the superior convergence rate and communication efficiency of our methods.

Index Terms—Multirobot systems, optimization, simultaneous localization and mapping (SLAM).

I. INTRODUCTION

COLLABORATIVE spatial perception is a fundamental capability for multirobot systems to operate in unknown, GPS-denied environments. State-of-the-art systems (e.g., [1], [2], [3], [4], [5], [6]) rely on optimization-based back-ends to achieve accurate multirobot simultaneous localization and mapping (SLAM). Often, a central server receives data from all robots (e.g., in the form of factor graphs [7]) and solves the underlying large-scale optimization for the entire team. In comparison, collaborative optimization frameworks leverage robots’ local computation and iterative communication (either peer-to-peer or coordinated by a server), and thus have the potential to scale to larger scenes and support more robots.

Recent works focus on developing fully distributed algorithms in which robots carry out iterative optimization via peer-to-peer message passing [8], [9], [10], [11], [12], [13]. While these methods are flexible in terms of the required communication architecture, they often suffer from slow convergence due to their first-order nature and the inherent poor conditioning of typical SLAM problems. To resolve the slow convergence issue, an alternative is to pursue a second-order optimization framework. A prominent example is DDF-SAM [14], [15], [16], in which robots marginalize out internal variables (i.e., those without interrobot measurements) in their local factor graphs before communication. From an optimization perspective, robots partially eliminate their local Hessians and communicate the resulting matrices.

However, a shortcoming of this approach is that the transmitted matrices are usually dense (even if the original problem is sparse), and hence could result in long transmission times that prevent the team from obtaining a timely solution.

To address the aforementioned technical gaps, this work presents results toward collaborative optimization that achieves both fast convergence and efficient communication. Specifically, we develop new algorithms for solving multirobot rotation averaging and translation estimation. These problems are fundamental and have applications ranging from initialization for pose graph SLAM [17], structure-from-motion (SfM) [18], and camera network localization [8]. Our approach is based on a server–client architecture [see Fig. 1(a)], in which multiple robots (clients) coordinate with a server to collaboratively solve the optimization problem leveraging local computation. The cruciality of our method lies in exploiting theoretical relations between the Hessians of the optimization problems and the Laplacians of the underlying graphs. We leverage these theoretical insights to develop a fast collaborative optimization method in which each iteration computes an approximate second-order update by replacing the Hessian with a constant Laplacian matrix, which improves efficiency in both computation and communication. Furthermore, during communication, robots use spectral sparsification [19], [20] to sparsify intermediate dense matrices resulted from elimination of its internal variables. Fig. 1(b)–(d) show a high-level illustration of our approach. By varying the degree of sparsification, our method thus provides a principled way for trading off accuracy with communication efficiency. The theoretical properties of spectral sparsification allow us to perform rigorous convergence analysis, and establish linear rates of convergence for our methods. Last, we also present an extension to outlier-robust estimation by combining our approach with graduated nonconvexity (GNC) [21], [22].

A. Contributions

The key contributions of this work are summarized as follows:
1) We present collaborative optimization algorithms for multirobot rotation averaging and translation estimation under the server–client architecture, which enjoy fast convergence (in terms of the number of iterations) and efficient communication through the use of spectral sparsification.

2) In contrast to the typical sublinear convergence of prior methods, we prove (local) linear convergence for our methods and show that the rate of convergence depends on the user-defined sparsification parameter.

3) We present an extension to outlier-robust estimation by combining the proposed algorithms with GNC.

4) We perform extensive evaluations of our methods and demonstrate their values on real-world SLAM and SFM scenarios with outlier measurements.

Last, while our algorithms and theoretical guarantees cover separate rotation averaging and translation estimation, we demonstrate through our experiments that their combination can be used to achieve robust initialization for pose graph optimization (PGO), which is another fundamental problem commonly used in collaborative SLAM.

B. Paper Organization

The rest of this article is organized as follows. The rest of this section introduces necessary notation and mathematical preliminaries, and in Section II, we review related works. Section III formally introduces the problem formulation, communication architecture, and relevant applications. In Section IV, we establish theoretical relations between the Hessians and the underlying graph Laplacians. Then, in Section V, we leverage these theoretical results to design fast and communication-efficient solvers for the problems of interest and establish convergence guarantees. Finally, Section VI presents numerical evaluations of the proposed algorithms. Proofs and additional details, discussions, and experiments are provided in our extended technical report [23].

C. Notations and Preliminaries

The detailed notations used in this work are summarized in [23; Table V]. Unless stated otherwise, lowercase and uppercase letters denote vectors and matrices, respectively. We define $[n] \triangleq \{1, 2, \ldots, n\}$ as the set of positive integers from 1 to $n$.

Linear Algebra and Spectral Approximation: $S^n$ and $S^n_+$ denote the set of $n \times n$ symmetric and symmetric positive semidefinite matrices, respectively. We use $\otimes$ to denote the Kronecker product. For a positive integer $n$, $1_n \in \mathbb{R}^n$ and $I_n \in \mathbb{R}^{n \times n}$ denote the vector of all ones and the Identity matrix. For any matrix $A$, $\ker(A)$ and $\text{image}(A)$ denote the kernel (nullspace) and image (span of column vectors) of $A$, respectively. $A^*$ denotes the Moore–Penrose inverse of $A$, which coincides with the inverse $A^{-1}$ when $A$ is invertible. When $A \in S^n$, $\lambda_1(A), \ldots, \lambda_n(A)$ denote the real eigenvalues of $A$ sorted in increasing order. When $A \in S^n_+$, we also define $||X||_A \triangleq \sqrt{\text{tr}(X^* A X)}$ where $X$ is of compatible dimensions.

Following [24], [25], for $A, B \in S^n$ and $\epsilon > 0$, we say that $B$ is an $\epsilon$-approximation of $A$, denoted as $A \approx_\epsilon B$, if the following holds:

$$e^{-\epsilon} B \preceq A \preceq e^\epsilon B$$

(1)

where $B \preceq A$ means $A - B \in S^n_+$. Note that $A^+$ is symmetric and holds under composition: if $A \approx_\epsilon B$ and $B \approx_\epsilon C$, then $A \approx_\epsilon C$. Furthermore, if $A$ is singular, the relation (1) implies that $B$ is necessarily singular and $\ker(A) = \ker(B)$.

Graph Theory: A weighted undirected graph is denoted as $G = (\mathcal{V}, \mathcal{E}, w)$, where $\mathcal{V}$ and $\mathcal{E}$ denote the vertex and edge sets, and $w : \mathcal{E} \rightarrow \mathbb{R}_{>0}$ is the edge weight function that assigns each edge $(i, j) \in \mathcal{E}$ a positive weight $w_{ij}$. For a graph $G$ with $n$ vertices, its graph Laplacian $L(G; w) \in S^n_+$ is defined as

$$L(G; w)_{ij} = \begin{cases} \sum_{k \in \text{Nbr}(i)} w_{ik}, & \text{if } i = j, \\ -w_{ij}, & \text{if } i \neq j, (i, j) \in \mathcal{E}, \\ 0, & \text{otherwise}. \end{cases}$$

(2)

In (2), $\text{Nbr}(i) \subseteq \mathcal{V}$ denotes the neighbors of vertex $i$ in the graph. Our notation $L(G; w)$ serves to emphasize that the Laplacian of $G$ depends on the edge weight $w$. When the edge weight $w$ is irrelevant or clear from context, we will write the graph as $G = (\mathcal{V}, \mathcal{E})$ and its Laplacian as $L(G)$ or simply $L$. The graph Laplacian $L$ always has a zero eigenvalue, i.e., $\lambda_1(L) = 0$. The second smallest eigenvalue $\lambda_2(L)$ is known as the algebraic connectivity, which is always positive for connected graphs.

Riemannian Manifolds: The reader is referred to [26] and [27] for a comprehensive review of optimization on manifold manifolds. In general, we use $M$ to denote a smooth matrix manifold. For integer $n > 1$, $M^n$ denotes the product manifold formed by $n$
copies of $\mathcal{M}$. $T_x\mathcal{M}$ denotes the tangent space at $x \in \mathcal{M}$. For tangent vectors $\eta, \xi \in T_x\mathcal{M}$, their inner product is denoted as $\langle \eta, \xi \rangle_x$, and the corresponding norm is $\|\eta\|_x = \sqrt{\langle \eta, \eta \rangle_x}$. In the rest of this article, we drop the subscript $x$ as it will be clear from context. At $x \in \mathcal{M}$, the injectivity radius $\text{inj}(x)$ is a positive constant such that the exponential map $\text{Exp}_x : T_x\mathcal{M} \to \mathcal{M}$ is a diffeomorphism when restricted to the domain $U = \{\eta \in T_x\mathcal{M} : \|\eta\| < \text{inj}(x)\}$. In this case, we define the logarithm map to be $\log_x \triangleq \text{Exp}_x^{-1}$. Unless otherwise mentioned, we use $d(x, y)$ to denote the geodesic distance between two points $x, y \in \mathcal{M}$ induced by the Riemannian metric. In addition, it holds that $d(x, y) = \|v\|$ where $v = \log_x (y)$; see [27, Proposition 10.22].

The Rotation Group $\text{SO}(d)$: The rotation group is denoted as $\text{SO}(d) = \{R \in \mathbb{R}^{d \times d} : R^2 = I, \det(R) = 1\}$. The tangent space at $R$ is given by $T_R\text{SO}(d) = \{RV : V \in \text{so}(d)\}$, where $\text{so}(d)$ is the space of $d \times d$ skew-symmetric matrices. In this work, we exclusively work with 2-D and 3-D rotations. We define a basis for $T_R\text{SO}(3)$ such that each tangent vector $\eta \in T_R\text{SO}(3)$ is identified with a vector $v \in \mathbb{R}^3$

$$\eta = R[v]_x = R \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}.$$  

(3)

Note that (3) defines a bijection between $\eta \in T_R\text{SO}(3)$ and $v \in \mathbb{R}^3$. For $d = 2$, we can define a similar basis for the 1-D tangent space $T_R\text{SO}(2)$, where each tangent vector $\eta \in T_R\text{SO}(2)$ is identified by a scalar $v \in \mathbb{R}$ as

$$\eta = R[v]_x = R \begin{bmatrix} 0 & -v \\ v & 0 \end{bmatrix}.$$  

(4)

We have overloaded the notation $[\cdot]_x$ to map the input scalar or vector to the corresponding skew-symmetric matrix in $\text{so}(2)$ or $\text{so}(3)$. Under the basis given in (3) and (4), the inner product on the tangent space is defined by the corresponding vector dot product, i.e., $\langle \eta_1, \eta_2 \rangle = v_1^T v_2$ where $v_1, v_2 \in \mathbb{R}^3$ are vector representations of $\eta_1$ and $\eta_2$, and $p = \dim \text{SO}(d) = (d(d-1))/2$. We define the function $\text{Exp} : \mathbb{R}^p \to \text{SO}(d)$ as

$$\text{Exp}(v) = \exp([v]_x).$$  

(5)

where $\exp(\cdot)$ denotes the conventional matrix exponential. Note that $\text{Exp} : \mathbb{R}^p \to \text{SO}(d)$ should not be confused with the exponential mapping on Riemannian manifolds $\text{Exp}_x : T_x\mathcal{M} \to \mathcal{M}$, although the two are closely related in the case of rotations. Specifically, at a point $R \in \text{SO}(d)$ where $d \in \{2, 3\}$, the exponential map can be written as $\text{Exp}_R(\eta) = R \text{Exp}(v)$. Last, we also denote $\log$ as the inverse of $\text{Exp}$ in (5).

II. RELATED WORKS

In this section, we review related work in collaborative SLAM (see Section II-A), graph structure on rotation averaging and PGO (see Section II-B), and the applications of spectral sparsification and Laplacian linear solvers (see Section II-C).

A. Collaborative SLAM

Systems: State-of-the-art collaborative SLAM systems rely on optimization-based back-ends to accurately estimate robots’ trajectories and maps in a global reference frame. In fully centralized systems (e.g., [1], [2], [3]), robots upload their processed measurements to a central server that in practice could contain e.g., odometry factors, visual keyframes, and/or lidar key frames. Using this information, the server is responsible for managing the multirobot maps and solving the entire back-end optimization problem. In contrast, in systems leveraging distributed computation (e.g., [4], [5], [6], [18], [28]), robots collaborate to solve back-end optimization by coordinating with a server or among themselves. The resulting communication usually involves exchanging intermediate iterates needed by distributed optimization to attain convergence.

Optimization Algorithms: To solve factor graph optimization in a multirobot setting, Cunningham et al. developed DDF-SAM [14, 16], where each agent communicates a “condensed graph” produced by marginalizing out internal variables (those without inter-robot measurements) in its local Gaussian factor graph. Researchers have also developed information-based sparsification methods to sparsify the dense information matrix after marginalization using Chow-Liu tree (e.g., [29], [30]) or convex optimization (e.g., [31], [32]). In these works, sparsification is guided by an information-theoretic objective such as the Kullback–Leibler divergence, and requires linearization to compute the information matrix. In comparison, our approach sparsifies the graph Laplacian that does not depend on linearization, and furthermore the sparsified results are used by collaborative optimization to achieve fast convergence.

From an optimization perspective, marginalization corresponds to a domain decomposition approach (e.g., see [33, Chap. 14]) where one eliminates a subset of variables in the Hessian using the Schur complement. Related works use sparse approximations of the resulting matrix (e.g., with tree-based sparsity patterns) to precondition the optimization [34, 35, 36, 37]. Recent work [28] combines domain decomposition with event-triggered transmission to improve communication efficiency during collaborative estimation.

Zhang et al. [38] developed a centralized incremental solver for multirobot SLAM. Fully decentralized solvers for SLAM have also gained increasing attention; see [8], [9], [10], [11], [12], [13], [39]. In the broader field of optimization, related works include decentralized consensus optimization methods such as [40], [41], [42], [43]. Compared to these fully decentralized methods, the proposed approach assumes a central server but achieves significantly faster convergence by implementing approximate second-order optimization.

B. Graph Structure in Rotation Averaging and PGO

Prior works have investigated the impact of graph structure on rotation averaging and PGO problems from different perspectives. One line of research [44, 45, 46] adopts an estimation-theoretic approach and shows that the Fisher information matrix is closely related to the underlying graph Laplacian matrix. Eriksson et al. [47] established sufficient conditions for strong duality to hold in rotation averaging, where the derived analytical error bound depends on the algebraic connectivity of the graph. Recently, Bernreiter et al. [48] used tools from graph signal processing to correct onboard estimation errors in multirobot mapping. Doherty et al. [49] proposed a measurement selection approach for pose graph SLAM that seeks to maximize the algebraic connectivity of the underlying graph. This article differs from the aforementioned works by analyzing the impact of graph structure on the underlying optimization problems, and
exploiting the theoretical analysis to design novel optimization algorithms in the multirobot setting.

Among related works in this area, the ones most related to this article are [50], [51], [52], [53], [54]. Carlone [50] analyzes the influences of graph connectivity and noise level on the convergence of Gauss–Newton methods when solving PGO. Tron [51] derives the Riemannian Hessian of rotation averaging under the geodesic distance, and uses the results to prove convergence of Riemannian gradient descent. Wilson et al. [52], [53] studied the local convexity of rotation averaging under the geodesic distance, by bounding the Riemannian Hessian using the Laplacian of a suitably weighted graph. Recently, Nasiri et al. [54] developed a Gauss–Newton method for rotation averaging under the chordal distance, and show that its convergence basin is influenced by the norm of the inverse reduced Laplacian matrix. Our work differs from [50], [51], [52], [53], [54] by focusing on the development of fast and communication-efficient solvers in multirobot teams with provable performance guarantees. During this process, we also prove new results on the connections between the Riemannian Hessian and graph Laplacian, and show that they hold under both geodesic and chordal distance.

C. Spectral Sparsification and Laplacian Solvers

A remarkable property of graph Laplacians is that they admit sparse approximations; see [19] for a survey. Spielman and Srivastava [20] show that every graph with \( n \) vertices can be approximated using a sparse graph with \( O(n \log n) \) edges. This is achieved using a random sampling procedure that selects each edge with probability proportional to its effective resistance, which intuitively measures the importance of each edge to the whole graph. Batson et al. [55] developed a procedure based on the so-called barrier functions for constructing linear-sized sparsifiers. Another line of work [56], [57] employs sparsification during approximate Gaussian elimination. Spectral sparsification is one of the main tools that enables recent progress in fast Laplacian solvers (i.e., for solving linear systems of the form \( Lx = b \), where \( L \) is a graph Laplacian); see [58] for a survey. Peng and Spielman [59] developed a parallel solver that invokes sparsification as a subroutine, which is improved and extended in following works [24], [25]. In this work, we leverage spectral sparsification to design communication-efficient collaborative optimization methods for rotation averaging with provable convergence guarantees.

III. PROBLEM FORMULATION

This section formally defines the rotation averaging and translation estimation problems in the multirobot context. For clarity, here we introduce the problems without considering outlier measurements, and present extensions to outlier-robust optimization in Section V-D. We review the communication and computation architectures used by our algorithms. Finally, we discuss relevant applications in multirobot SLAM and SfM.

A. Rotation Averaging

We model rotation averaging using an undirected measurement graph \( G = (V, E) \). Each vertex \( i \in V = [n] \) corresponds to a rotation variable \( R_i \in \text{SO}(d) \) to be estimated. Each edge \( (i, j) \in E \) corresponds to a noisy relative measurement of the form

\[
\tilde{R}_{ij} = R_i^T R_j R_{ij}^{\text{ref}}
\]

(6)

where \( R_i, R_j \in \text{SO}(d) \) are the latent (ground truth) rotations and \( R_{ij}^{\text{ref}} \in \text{SO}(d) \) is the measurement noise. In standard rotation averaging, we aim to estimate the rotations by minimizing the sum of squared measurement residuals, which corresponds to the formulation in Problem 1.

Problem 1 (Rotation Averaging):

\[
\begin{aligned}
\text{minimize} \\
R = (R_1, \ldots, R_n) \in \text{SO}(d)^n \\
\sum_{(i, j) \in E} \kappa_{ij} \varphi(R_i \tilde{R}_{ij}, R_j).
\end{aligned}
\]

(7)

For each edge \( (i, j) \in E \), \( \kappa_{ij} > 0 \) is the corresponding measurement weight. The function \( \varphi \) is defined as either the squared geodesic (8a) or chordal distance (8b).

\[
\varphi(R_i \tilde{R}_{ij}, R_j) \triangleq \begin{cases} \\
\frac{1}{2} \| \log(R_i \tilde{R}_{ij} R_j^T) \|_F^2, \\
\frac{1}{2} \| R_i \tilde{R}_{ij} - R_j \|_F^2.
\end{cases}
\]

(8a)

(8b)

In the multirobot setting, each robot owns a subset of all rotation variables and only knows about measurements involving its own variables; see Fig. 1(b) for an illustration.

B. Translation Estimation

Similar to rotation averaging, we also consider the problem of estimating multiple translation vectors given noisy relative translation measurements.

Problem 2 (Translation Estimation):

\[
\begin{aligned}
\text{minimize} \\
l = (t_1, \ldots, t_n) \in \mathbb{R}^{d \times n} \\
\sum_{(i, j) \in E} \frac{\tau_{ij}}{2} \| t_j - t_i - \tilde{t}_{ij} \|_2^2.
\end{aligned}
\]

(9)

Note that (9) is a linear least squares problem. Similar to rotation averaging, (9) can be modeled using the undirected measurement graph \( G = (V, E) \), where vertex \( i \) represents the translation variable \( t_i \in \mathbb{R}^d \) to be estimated, and edge \( (i, j) \in E \) represents the relative translation measurement \( \tilde{t}_{ij} \in \mathbb{R}^d \). Last, \( \tau_{ij} > 0 \) is the positive weight associated with measurement \( (i, j) \in E \).

C. Communication and Computation Architecture

In this work, we consider solving Problems 1 and 2 under the server–client architecture. As shown in Fig. 1(a), a central server coordinates with all robots (clients) to solve the overall problem by distributing the underlying computation to the entire team. In practice, the server could itself be a robot (e.g., in multirobot exploration scenarios) or a remote machine (e.g., in cloud-based AR/VR applications). Each iteration (communication round) consists of an upload stage in which robots perform parallel local computations and transmit their intermediate information to the server, and a download stage in which the server aggregates information from all robots and broadcasts back the result. When a direct communication link to the server does not exist, a robot can still participate in this framework by relaying its information through other robots. By leveraging local computations, the server–client architecture can scale better compared to a fully centralized approach in which the server solves the entire optimization problem. At the same time, by implementing second-order optimization algorithms, this architecture also produces
of rotation averaging: in (7), note that left multiplying each rotation \( R_i = \text{SO}(d) \), \( i \in [n] \) by a common rotation \( S \in \text{SO}(d) \) does not change the cost function. As a result, each solution \( R = (R_1, \ldots, R_n) \in \text{SO}(d)^n \) actually corresponds to an equivalence class of solutions in the form of
\[
[R] = \{ (SR_1, \ldots, SR_n) \mid S \in \text{SO}(d) \}. \tag{11}
\]

The equivalence relation (11) shows that rotation averaging is actually an optimization problem defined over a quotient manifold \( M = \mathbb{R}^n \sim \), where \( \mathbb{R}^n \) is the total space and \( \sim \) denotes the equivalence relation underlying (11); see [27, Chap. 9] for more details. Accounting for the quotient structure is critical for establishing the relation between the Hessian and the graph Laplacian.

In this work, we are interested in applying Newton’s method on the quotient manifold \( M \) due to its superior convergence rate. The Newton update can be derived by considering a local perturbation of the cost function. Specifically, let \( R = (R_1, \ldots, R_n) \in \text{SO}(d)^n \) be our current rotation estimates. For each rotation matrix \( R_i \), we seek a local correction to it in the form of \( \text{Exp}(v_i)R_i \), where \( v_i \in \mathbb{R}^d \) is some vector to be determined and \( \text{Exp}(\cdot) \) is defined in (5). In (7), replacing each \( R_i \) with its correction \( \text{Exp}(v_i)R_i \) leads to the following local approximation\(^1\) of the optimization problem
\[
\min_{v \in \mathbb{R}^{pn}} h(v; R) \triangleq \sum_{(i,j) \in E} \kappa_{ij} \varphi(\text{Exp}(v_i)R_i \tilde{t}_{ij}, \text{Exp}(v_j)R_j).
\tag{12}
\]
In (12), the overall vector \( v \in \mathbb{R}^{pn} \) is formed by concatenating all \( v_i \)'s. Compared to (7), the optimization variable in (12) becomes the vector \( v \) and the rotations \( R \) are treated as fixed. Furthermore, we note that the quotient structure of Problem 1 gives rise to the following vertical space [27, Chap. 9.4] that summarizes all directions of change along which (12) is invariant
\[
\mathcal{N} = \text{image}(1_n \otimes I_p) \subset \mathbb{R}^{pn}. \tag{13}
\]

Intuitively, \( \mathcal{N} \) captures the action of any global rotation. Indeed, for any \( v \in \mathcal{N} \), we have \( \text{Exp}(v_i) = \text{Exp}(v_j) \) for all \( i, j \in [n] \), and thus the cost function (12) remains constant. Let us denote the gradient and Hessian of (12) as follows:
\[
\bar{g}(R) \triangleq \nabla h(v; R)|_{v=0}, \quad \bar{H}(R) \triangleq \nabla^2 h(v; R)|_{v=0}. \tag{14}
\]
Our notations \( \bar{g}(R) \) and \( \bar{H}(R) \) serve to emphasize that the gradient and Hessian are defined in the total space \( \mathbb{R}^n \) and depend on the current rotation estimates \( R \). Let \( \mathcal{H} \triangleq \mathcal{N}^\perp \) denote the horizontal space, which is the orthogonal complement of the vertical space \( \mathcal{N} \). In [27, Chap. 9.12], it is shown that executing the Newton update on the quotient manifold amounts to finding the solution \( v \in \mathcal{H} \) to the linear system
\[
(P_T \bar{H}(R) P_T)\nu = -\bar{g}(R)\tag{15}
\]
where \( P_T \) is the orthogonal projection onto \( \mathcal{H} \). We note that \( P_T \) is symmetric, and so is \( \bar{H}(R) \). Furthermore, it holds that

\(^1\)The approximation defined in (12) is closely related to the standard pullback function in Riemannian optimization; see [23, Appendix II-D]. In this work, we use (12) since the resulting Hessian has a particularly interesting relationship with the graph Laplacian matrix, as shown in Theorem 1.
\( \overline{\mathcal{F}}(R) = P_H \overline{\mathcal{F}}(R) \), which follows from known results on optimization over quotient manifolds (see Remark 2 for details). Intuitively, including \( P_H \) in (15) accounts for the gauge symmetry by eliminating the effect of any vertical component from \( v \). The following theorem reveals an interesting connection between \( H(R) \) defined in (15) and the Laplacian of the underlying graph.

**Theorem 1 (Local Hessian Approximation for Rotation Averaging):** Let \( \overline{R} \in \text{SO}(d)^n \) denote the set of ground truth rotations from which the noisy measurements \( \overline{R}_{ij} \) are generated according to (6). For any \( \delta \in (0, 1/2) \), there exist constants \( \theta, r > 0 \) such that

\[
d(\overline{R}_{ij}, \overline{R}_i^T \overline{R}_j) \leq \theta, \forall (i, j) \in \mathcal{E}
\]

then for all \( R \in B_r(R^*) = \{ R \in \text{SO}(d)^n : d(R, R^*) < r \} \) where \( R^* \in \text{SO}(d) \) is a global minimizer of Problem 1, it holds that

\[
H(R) \approx_{\delta} L(G; w) \otimes I_p.
\]

In (17), \( G = (V, \mathcal{E}) \) is the measurement graph, and \( p = \dim \text{SO}(d) \). For edge \((i,j) \in \mathcal{E}\), its edge weight \( w_{ij} \) is given by \( w_{ij} = \kappa_{ij} \) for the squared geodesic distance cost (8a), and \( w_{ij} = 2\kappa_{ij} \) for the squared chordal distance cost (8b).

Before proceeding, we note that Theorem 1 directly implies the following bound on the Hessian \( H(R) \).

**Corollary 1 (Local Hessian Bound and Condition Number for Rotation Averaging):** Under the assumptions of Theorem 1, define constants \( \mu_H = e^{-\delta} \lambda_2(L(G; w)) \) and \( L_H = e^{\delta} \lambda_n(L(G; w)) \). Then, for all \( R \in B_r(R^*) \)

\[
\mu_H P_H \preceq H(R) \preceq L_H P_H.
\]

In the following, \( \kappa_H = L_H / \mu_H \) is referred to as the condition number.

We prove Theorem 1 and Corollary 1 in [23, Appendix II]. Theorem 1 shows that under small measurement noise, the Hessian near a global minimizer is well approximated by the Laplacian of an appropriately weighted graph.\(^2\) In Fig. 2, we perform numerical validation of this result using synthetic chordal rotation averaging problems defined over a 3-D grid with 125 rotation variables [see Fig. 2(a)]. With a probability of 0.3, we generate noisy relative measurements between pairs of nearby rotations, corrupted by increasing levels of Langevin noise [60, Appendix A]. At each noise level, we obtain the global minimizer \( R^* \) (global optimality is certified using the approach in [47]) and numerically compute the smallest constant \( \delta \) such that \( H(R^*) \approx_{\delta} L \otimes I_p \). Fig. 2(b) shows the evolution of the empirical approximation constant \( \delta \) as a function of noise level. In the special case when there is no noise, it can be shown that \( H(R^*) = L \otimes I_p \), and thus the empirical \( \delta \) is zero. In general, the empirical value of \( \delta \) increases smoothly as the noise level increases. Since the Hessian \( H(R) \) varies smoothly with \( R \), our results confirm that the Laplacian is a good approximation of the Hessian locally around \( R^* \), as predicted by Theorem 1.

\(^2\)Currently, Theorem 1 only shows the existence of constants \( \theta, r > 0 \) such that the approximation relation (17) holds. In a nutshell, this is because our proof is based on the following key relation that holds in the limit: if we define \( \theta_{ij}(R) = d(\overline{R}_{ij}, \overline{R}_i^T \overline{R}_j) \) as the measurement residual of edge \((i,j) \in \mathcal{E} \) at a solution \( R \in \text{SO}(d)^n \), then we can show that \( H(R) \rightarrow L(G; w) \otimes I_p \) as \( \theta_{ij}(R) \rightarrow 0 \) for all \((i,j) \in \mathcal{E} \); see discussions around (98) in [23, Appendix II]. While it would be interesting to derive explicit and accurate bounds for \( \theta \) and \( r \) (as a function of \( \delta \)), this would require a substantial improvement to our current proof technique, which we leave for future work.

The result in Theorem 1 directly motivates an approximate Newton method that replaces the Hessian with its Laplacian approximation. Specifically, instead of solving (15), one solves the following approximate Newton system

\[
(L(G; w) \otimes I_p) v = -\overline{\mathcal{F}}(R).
\]

In the following, it would be more convenient to consider the matrix form of the above linear system. For this purpose, let us define matrices \( V, B(R) \in \mathbb{R}^{n \times p} \)

\[
V \triangleq \begin{bmatrix} v_1^{\top} \\ \vdots \\ v_n^{\top} \end{bmatrix}, \quad B(R) \triangleq \begin{bmatrix} -\overline{\mathcal{F}}_1(R)^{\top} \\ \vdots \\ -\overline{\mathcal{F}}_n(R)^{\top} \end{bmatrix}.
\]

Using properties of the Kronecker product, we can show that (19) is equivalent to

\[
L(G; w) V = B(R).
\]

Algorithm 1 shows the pseudocode of the approximate Newton algorithm. Compared to the original Newton’s method, Algorithm 1 uses a constant matrix across all iterations, and hence could be significantly more computationally efficient by avoiding to recompute and refactorize the Hessian matrix at every iteration. For this reason, we believe that Algorithm 1 could be of independent interest for standard (centralized) rotation averaging. Furthermore, in Section V, we show that Algorithm 1 admits communication-efficient extensions in multirobot settings.

**Remark 1 (Connections with prior work):** Theorem 1 leverages prior theories developed by Tron [51] and Wilson et al. [52, 53] and extend them to cover rotation averaging under both geodesic and chordal distance metrics. Nasiri et al. [54] first developed Algorithm 1 for chordal rotation averaging using a Gauss–Newton formulation. In contrast, we motivate Algorithm 1 by proving the theoretical approximation relation between the Hessian and the graph Laplacian (see Theorem 1). Last, the theoretical approximation relation we establish also allows us to prove local linear convergence for our methods.

**Remark 2 (Feasibility of the approximate Newton system):** Using the properties of the graph Laplacian and the Kronecker product, we see that \( \ker(L(G; w) \otimes I_p) = \mathcal{N} \), where

![Image](image318x630 to 405x725)

Fig. 2. Empirical validation of the Hessian approximation relation in Theorem 1. (a) Example synthetic chordal rotation averaging problem with 125 rotations. Each rotation is visualized as an oriented camera. Each blue edge shows a relative rotation measurement corrupted by Langevin noise. (b) Evolution of the empirical approximation constant \( \delta \) such that \( H(R^*) \approx_{\delta} L \otimes I_p \). We perform 20 runs for each noise level. Solid line denotes the average value for \( \delta \) and the surrounding shaded area shows one standard deviation. (a) Grid simulation. (b) Empirical values of \( \delta \).
Algorithm 1: Approximate Newton’s method for rotation averaging.

1: for iteration $k = 0, 1, \ldots$ do
2: Compute approximate Newton update by solving $L(G; w) V^k = B(R^k)$.
3: Update iterate by $R^{k+1} = \text{Exp}(v^k)^k R^k$, for all $i \in [n]$.
4: end for

$N$ is the vertical space defined in (13). Furthermore, in [27, Chap. 9.8], it is shown that $\pi(R) \perp N$. Thus, we conclude that $\pi(R) \in \text{image}(L(G; w) \otimes I_p)$, i.e., the linear system (19) and its equivalent matrix form (21) are always feasible. In fact, the system is singular and hence admits infinitely many solutions. Similar to the original Newton’s method on quotient manifold, we will select the minimum norm solution $v$, which guarantees that $v \in H$ [27, Chap. 9.12].

B. Translation Estimation

Unlike rotation averaging, translation estimation (Problem 2) is a convex linear least squares problem. In particular, it can be shown that Problem 2 is equivalent to a linear system involving the graph Laplacian $L(G; \tau)$, where $\tau : \mathcal{E} \to \mathbb{R}_{>0}$ is the edge weight function that assigns each edge $(i, j) \in \mathcal{E}$ a weight given by the corresponding translation measurement weight $\tau_{ij}$ in Problem 2. Denote $M_t = [t_1 \ldots t_n]^T \in \mathbb{R}^{n \times d}$ as the matrix where each row corresponds to a translation vector to be estimated. One can show that the optimal translations are solutions of

$$L(G; \tau) M_t = B_t$$

(22)

where $B_t \in \mathbb{R}^{n \times d}$ is a constant matrix that only depends on the measurements. Furthermore, each column of $B_t$ belongs to the image of the Laplacian $L(G; \tau)$, so (22) is always feasible; see [60, Appendix B.2] for details. To conclude this section, we note that similar to rotation averaging, translation estimation (Problem 2) is subject to a gauge symmetry. Specifically, two translation solutions $M_t$ and $M_t'$ are equivalent if they only differ by a global translation. Mathematically, this means that $M_t = M_t' + 1_n t_0$, where $1_n \in \mathbb{R}^n$ is the vector of all ones and $t_0 \in \mathbb{R}^d$ is the constant global translation vector.

V. ALGORITHMS AND PERFORMANCE GUARANTEES

In Section IV, we have shown that Laplacian systems naturally arise when solving the rotation averaging and translation estimation problems; see (21) and (22), respectively. Recall that we seek to find the solution $X \in \mathbb{R}^{n \times p}$ to a linear system of the form

$$LX = B$$

(23)

where $L \in \mathbb{S}^n_+$ is the Laplacian of the multirobot measurement graph [see Fig. 1(b)], and each column of $B \in \mathbb{R}^{n \times p}$ is in the image of $L$ so that (23) is always feasible. For rotation averaging, we have $p = \dim \text{SO}(d)$, and for translation estimation, we have $p = \dim \mathbb{R}^d = d$. In Section V-A, we develop a communication-efficient solver for (23) under the server–client architecture described in Section III-C. Then, in Sections V-B and V-C, we use the developed solver to design communication-efficient algorithms for collaborative rotation averaging and translation estimation, and establish convergence guarantees for both cases. Last, in Section V-D, we present extension to outlier-robust estimation based on GNC.

A. Collaborative Laplacian Solver With Spectral Sparsification

We propose to solve (23) using the domain decomposition framework [33, Chap. 14], which has been utilized in earlier works such as DDF-SAM [14, 15, 16] to solve collaborative SLAM problems. This is motivated by the fact that in the multirobot measurement graph with $m$ robots, there is a natural disjoint partitioning of the vertex set $\mathcal{V}$

$$\mathcal{V} = \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_m$$

(24)

where $\mathcal{V}_i$ contains all vertices (variables) of robot $\alpha \in [m]$ and $\mathcal{V}$ denotes the disjoint union. Furthermore, $\mathcal{V}_i$ can be partitioned as $\mathcal{V}_i = \mathcal{F}_\alpha \cup \mathcal{C}_\alpha$, where $\mathcal{C}_\alpha$ denotes all separator (interface) vertices and $\mathcal{F}_\alpha$ denotes all interior vertices of robot $\alpha$. In multirobot SLAM, the separators are given by the set of variables that have inter-robot measurements; see Fig. 1(b). Note that given the set of all separators $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_m$, robots’ interior vertices $\mathcal{F}_\alpha$ become disconnected from each other. The natural vertex partitioning in (24) further gives rise to a disjoint partitioning of the edge set

$$\mathcal{E} = \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_m \cup \mathcal{E}_c.$$  

(25)

For each robot $\alpha \in [m]$, its local edge set $\mathcal{E}_\alpha$ consists of all edges that connect two vertices from $\mathcal{V}_\alpha$. In Fig. 1(b), the local edges are shown using colors corresponding to the robots. The remaining inter-robot edges form $\mathcal{E}_c$, which are highlighted as bold black edges in Fig. 1(b).

In domain decomposition, we adopt a variable ordering in which the interior nodes $\mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_m$ appear before the separators $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_m$. With this variable ordering, the Laplacian system (23) can be rewritten as

$$\begin{bmatrix} L_{11} & L_{1e} & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ L_{me} & L_{ee} & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ L_{em} & L_{me} & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \ddots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & B_e \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_m \\ \vdots \\ X_e \end{bmatrix} = \begin{bmatrix} B_1 \\ \vdots \\ B_m \\ \vdots \\ B_e \end{bmatrix}.$$  

(26)

For $\alpha \in [m]$, $X_\alpha$ and $B_\alpha$ denote the rows of $X$ and $B$ in (23) that correspond to robot $\alpha$’s interior variables $\mathcal{F}_\alpha$. On the other hand, we treat separators from all robots as a single block $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_m$. In (26), we use the subscript $e$ to index rows and columns of matrices that correspond to $\mathcal{C}$.

Remark 3 (Computation of (26) under the server–client architecture): Under the server–client architecture we consider, the overall Laplacian system (26) is stored distributedly across the robots (clients) and the server. Specifically, since each robot $\alpha$ knows the subgraph induced by its own vertices $\mathcal{V}_\alpha$ [e.g., in Fig. 1(b), robot 2 knows all edges incident to the blue vertices], it independently computes and stores its Laplacian blocks $L_{\alpha e}$ and $L_{ee}$. Similarly, each robot $\alpha$ also independently computes and stores the block $B_\alpha$. Meanwhile, we assume that the blocks defined over separators $L_{ee}$ and $B_e$ are handled by the central server that performs additional computations.

In (26), the special “arrowhead” sparsity pattern motivates us to first solve the reduced system defined over the separators,
obtained by eliminating all interior nodes using the Schur complement [33, Chap. 14.2]

\[
\begin{bmatrix}
L_{cc} - \sum_{\alpha \in [m]} L_{ca}L_{a\alpha}^{-1}L_{ac}
\end{bmatrix} = \begin{bmatrix}
X_c = B_c - \sum_{\alpha \in [m]} L_{ca}L_{a\alpha}^{-1}B_a.
\end{bmatrix}
\]

In the following, let us define \( U_\alpha \triangleq L_{ca}L_{a\alpha}^{-1}B_a \) for each robot \( \alpha \in [m] \). Then, the matrix on the right-hand side of (27) can be written as

\[
U \triangleq B_c - \sum_{\alpha \in [m]} U_\alpha.
\] (28)

Meanwhile, the matrix \( S \) defined on the left-hand side of (27) is the Schur complement resulting from eliminating all interior nodes \( \mathcal{F} \) from the full Laplacian matrix \( L \), denoted as \( S = \text{Sc}(L, \mathcal{F}) \). The next lemma shows \( S \) is the sum of multiple smaller Laplacian matrices.

**Lemma 1:** For each robot \( \alpha \in [m] \), define \( G_\alpha = (\mathcal{F}_\alpha \cup \mathcal{C}, \mathcal{E}_\alpha) \) as its local graph induced by its interior edges \( \mathcal{E}_\alpha \). Let \( S_\alpha \) be the matrix resulting from eliminating robot \( \alpha \)'s interior vertices \( \mathcal{F}_\alpha \) from the Laplacian of \( G_\alpha \), i.e., \( S_\alpha = \text{Sc}(L(G_\alpha), \mathcal{F}_\alpha) \). Furthermore, define \( G_c = (\mathcal{C}, \mathcal{E}_c) \) as the graph induced by inter-robot loop closures \( \mathcal{E}_c \). Then, the matrix \( S \) that appears in (27) can be written as

\[
S = L(G_c) + \sum_{\alpha \in [m]} S_\alpha.
\] (29)

Lemma 1 is proved in [23, Appendix III-A]. Since Laplacian matrices are closed under Schur complements [24, Fact 4.2], each \( S_\alpha \) defined in Lemma 1 is also a Laplacian matrix. Furthermore, as a result of Remark 3, each robot \( \alpha \) can independently compute \( S_\alpha = \text{Sc}(L(G_\alpha), \mathcal{F}_\alpha) \) and upload \( U_\alpha = L_{ca}L_{a\alpha}^{-1}B_a \) to the server in parallel. Upon collecting \( S_\alpha \) and \( U_\alpha \) from all robots, the server can then solve (29) and (28). It then solves the linear system \( SX_c = U \) (27) and broadcasts the solution \( X_c \) back to all robots. Finally, once robots receive the separator solutions \( X_c \), they can in parallel recover their interior solutions via back-substitution,

\[
X_\alpha = L_{a\alpha}^{-1}(B_a - L_{ac}X_c).
\] (30)

The aforementioned method is a multirobot implementation of domain decomposition. While it effectively exploits the separable structure in the problem, this method can incur significant communication cost as it requires each robot \( \alpha \) to transmit its Schur complement matrix \( S_\alpha \) that is potentially dense. This issue is illustrated in Fig. 1(c), where for robot 2 (blue) its \( S_\alpha \) corresponds to a dense graph over its separators.

In the following, we propose an approximate domain decomposition algorithm that is significantly more communication-efficient while providing provable approximation guarantees. Our method is based on the facts that (i) each local Schur complement \( S_\alpha \) is itself a graph Laplacian, and (ii) graph Laplacians admit spectral sparsifications [19], i.e., for a given approximation threshold \( \epsilon > 0 \), one can compute a sparse Laplacian \( \tilde{S}_\alpha \) such that \( \tilde{S}_\alpha \approx \epsilon S_\alpha \). Generally, a larger value of \( \epsilon \) leads to a sparser \( \tilde{S}_\alpha \). In this work, we implement the method of Spielman and Srivastava [20] that sparsifies \( S_\alpha \) by sampling edges in the corresponding dense graph based on their effective resistances. Intuitively, the effective resistances measure the importance of edges to the overall graph connectivity. The sparse matrix \( \tilde{S}_\alpha \) produced by this method has \( O(|\mathcal{C}| \log |\mathcal{C}|) \) entries, as opposed to the worst case \( O(|\mathcal{C}|^2) \) entries in \( S_\alpha \). We provide the complete description and pseudocode of the sparsification algorithm in [23, Appendix I]. Fig. 1(d) illustrates a spectral sparsification for robot 2’s dense reduced graph. In the proposed method, each robot transmits its sparse approximation \( \tilde{S}_\alpha \) instead of the original Schur complement \( S_\alpha \). By summing together these \( \tilde{S}_\alpha \) matrices, the server can obtain a sparse approximation to the original dense Schur complement \( S \); see Algorithm 2. Then, we can follow the same procedure as standard domain decomposition to obtain an approximate solution to the Laplacian system (23); see Algorithm 3. Specifically, the server first solves an approximate reduced system using \( \tilde{S} \) obtained from Algorithm 2 (line 7). Then, the interior solution for each robot is recovered using back-substitution (line 9).

Together, Algorithms 2 and 3 provide a parallel procedure for computing an approximate solution to the original Laplacian system (23) in the server–client architecture. Crucially, the use of spectral sparsifiers allows us to establish theoretical guarantees on the accuracy of the approximate solution as stated in the following theorem.

**Theorem 2 (Approximation guarantees of Algorithms 2 and 3):** Given a Laplacian system \( LX = B \), Algorithms 2 and 3 together return a solution \( \tilde{X} \in \mathbb{R}^{n \times p} \) such that \( \tilde{L} \tilde{X} = \tilde{B} \), where \( \tilde{L} \in S^m_+ \) satisfies

\[
\tilde{L} \approx \epsilon L.
\] (31)

---

**Algorithm 2: Sparsified Schur complement.**

1: function \( \tilde{S} = \text{SparsifiedSchurComplement}L, \epsilon \)
2: for each robot \( \alpha \) in parallel do
3: Compute a sparse approximation \( \tilde{S}_\alpha \) such that \( \tilde{S}_\alpha \approx \epsilon S_\alpha \).
4: Upload \( \tilde{S}_\alpha \) to the server.
5: end for
6: Server computes and stores \( \tilde{S} = L(G_c) + \sum_{\alpha \in [m]} \tilde{S}_\alpha \).
7: end function

**Algorithm 3: Sparsified Laplacian solver.**

1: function \( X = \text{SparsifiedLaplacianSolver}L, B, \tilde{S} \)
2: for each robot \( \alpha \) in parallel do
3: Compute \( U_\alpha = L_{ca}L_{a\alpha}^{-1}B_a \).
4: Upload \( U_\alpha \) to the server.
5: end for
6: Server collects \( U_\alpha \) and computes \( U = B_c - \sum_{\alpha \in [m]} U_\alpha \).
7: Server solves \( SX_c = U \) (where \( \tilde{S} \) is obtained from Algorithm 2), and broadcasts solution \( X_c \) to all robots.
8: for each robot \( \alpha \) in parallel do
9: Compute interior solution \( X_\alpha = L_{a\alpha}^{-1}(B_a - L_{ac}X_c) \).
10: end for
11: end function
Furthermore, let $X^* \in \mathbb{R}^{n \times p}$ be an exact solution to the input linear system, i.e., $LX^* = B$. It holds that

$$
\left\| \bar{X} - X^* \right\|_L \leq c(\epsilon) \left\| X^* \right\|_L
$$

(32)

where the constant $c(\epsilon)$ is defined as

$$
c(\epsilon) = \sqrt{1 + e^{2\epsilon}} - 2e^{-\epsilon}.
$$

We prove Theorem 2 in [23, Appendix III-B]. We have shown that the approximate solution $\bar{X}$ produced by Algorithms 2 and 3 remains close to the exact solution $X^*$ when measured using the “norm” induced by the original Laplacian $L$. Furthermore, the quality of the approximation is controlled by the sparsification parameter $\epsilon$ through the function $c(\epsilon)$ visualized in Fig. 3. Note that when $\epsilon = 0$, sparsification is effectively skipped and robots transmit the original dense matrices $S_\alpha$. In this case, we have $c(\epsilon) = 0$ and the solution $\bar{X}$ produced by our methods is exact, i.e., $L\bar{X} = B$. Meanwhile, by increasing $\epsilon$, our methods smoothly tradeoff accuracy with communication efficiency.

**Remark 4 (Connections with existing Laplacian solvers [24, 25]):** Our collaborative Laplacian solver (Algorithms 2 and 3) is inspired by the centralized solvers developed in [24, 25] for solving Laplacian systems in nearly linear time. However, our result differs from these works by focusing on the use of spectral sparsification in the multirobot setting to achieve communication efficiency. Furthermore, in Section V-B, we apply our Laplacian solver on the nonconvex Riemannian optimization problem underlying rotation averaging, and establish provable convergence guarantees for the resulting Riemannian optimization algorithm.

**Remark 5 (Communication efficiency of Algorithms 2 and 3):** We discuss the communication costs of Algorithms 2 and 3 under the server–client architecture. Denote the number of separators in the measurement graph as $|C|$. In Algorithm 2, each robot uploads the sparsified matrix $\bar{S}_\alpha$ to the server (line 4), which is guaranteed to have $O(|C| \log |C|)$ entries [20]. Consequently, Algorithm 2 incurs a total upload cost of $O(m|C| \log |C|)$, where $m$ is the number of robots. In Algorithm 3, robots upload their block vectors $U_\alpha$ in parallel (line 4) and the server broadcasts back the solution $X_\epsilon$ (line 7). Since both $U_\alpha$ and $X_\epsilon$ have a dimension of $|C|$-by-$p$ (where $p$ is constant), Algorithm 3 uses $O(m|C|)$ communication in both upload and download stages.

### B. Collaborative Rotation Averaging

In this section, we utilize the Laplacian solver developed in the previous section to design a fast and communication-efficient solver for rotation averaging. Recall the centralized method in Algorithm 1, where each iteration solves a Laplacian system $LV = B(R)$. In the multirobot setting, we can use the solver developed in Section V-A to obtain an approximate solution to this system. Algorithm 4 shows the pseudocode. First, an initial guess $R^0$ is computed (line 1). Then, at line 2, robots first form the approximate Schur complement $\bar{S}$ using SPARSIFIEDCHURCOMPLEMENT (Algorithm 2). Each iteration consists of three main steps. At the first step (line 4–8), robots compute and store the right-hand side $B(R)$, specifically, recall from Remark 3 that the overall $B(R)$ is divided into multiple blocks,

$$
B(R) = \left[ B(R)^\top_1 \cdots B(R)^\top_m B(R)_{m+1}^\top \right].
$$

(34)

In our algorithm, each robot $\alpha \in [m]$ computes the block $B(R)_{\alpha}$ corresponding to its interior variables $F_{\alpha}$, and the server computes the block $B(R)_{\epsilon}$ corresponding to all separators. At the second step (line 10), robots collaboratively solve for the update vector $V^k$ by calling SPARSIFIEDCHURSOLVER (Algorithm 3). Finally, at the last step (line 11–14), we obtain the next iterate using the solutions $V^k$, where robots in parallel update the rotation variables they own.

In the following, we proceed to establish theoretical guarantees for our collaborative rotation averaging algorithm. We will show that starting from a suitable initial guess, Algorithm 4 converges to a global minimizer at a linear rate. One might be tempted to state the linear convergence result on the total space, i.e., $d(R^{k+1}, R^*) \leq \gamma d(R^k, R^*)$, where $k$ is the iteration number, $\gamma \in (0, 1)$ is a constant, and $R^*$ is a global minimizer. However, it is challenging to prove this statement due to the gauge symmetry of rotation averaging. The iterates $\{R^k\}$ might converge to a solution $R^\infty$ that is only equivalent to $R^*$ up to a global rotation, i.e.,

$$
(SR^\infty_1, \ldots, SR^\infty_n) = (R_1^*, \ldots, R_n^*), \quad \text{for some } S \in SO(d).
$$

(35)

and as a result $d(R^\infty, R^*) \neq 0$ in general. Fortunately, this issue can be resolved using the machinery of Riemannian quotient manifolds. Instead of measuring the distance on the total space $d(R^k, R^*)$, we will compute the distance between the underlying equivalence classes $d([R^k], [R^*])$. We note that

---

**Algorithm 4:** Collaborative rotation averaging.

1: Initialize rotation estimates $R^0$.
2: $\bar{S} = \text{SPARSIFIEDCHURCOMPLEMENT}(L, \epsilon)$.
3: for iteration $k = 0, 1, \ldots$
4: // Distributed computation of $B(R^k)$
5: Server computes $B(R^k)_{\epsilon}$ that corresponds to all separators.
6: for each robot $\alpha$ in parallel do
7: Compute $B(R^k)_{\alpha}$ that corresponds to interior $F_{\alpha}$.
8: // Single round of communication to compute $V^k$
9: Solve
10: $V^k = \text{SPARSIFIEDCHURSOLVER}(L, B(R^k), \bar{S})$.
11: // Distributed updates of all rotation variables
12: for each robot $\alpha$ in parallel do
13: Update iterates by $R_{\alpha}^{k+1} = \text{Exp}(v^k)R^{k}_{\alpha}$, for each rotation variable $R_{\alpha}$ owned by robot $\alpha$.
14: end for
15: end for
\[
H(R)v = -\overline{g}(R) \\
H(R) \approx \delta L \otimes I_p \\
(L \otimes I_p)v = -\overline{g}(R) \\
L \approx \tilde{L} \\
(\tilde{L} \otimes I_p)v = -\overline{g}(R)
\]

Fig. 4. Intuitions behind the convergence rate in Theorem 3. Recall from Theorem 1 that under bounded measurement noise, the original Newton system (left box) is locally \(\delta\)-approximated by a linear system specified by a Laplacian \(L\) (middle box). In addition, in Theorem 2 we have shown that our distributed Laplacian solver approximates \(L\) with \(L \approx \epsilon \tilde{L}\) (right box). The composition of the two approximation relations thus gives \(H(R) \approx \delta + \epsilon (L \otimes I_p)\), which intuitively explains why (36) depends on a function of \(\delta + \epsilon\).

The distance metric \(d([R^k], [R^\ell])\) is well-defined since a quotient manifold inherits the Riemannian metric from its total space. Theorem 3 states the convergence result for Algorithm 4.

**Theorem 3 (Convergence rate of Algorithm 4):** Define \(\gamma(x) = 2\sqrt{\kappa_H} c(x)\) where \(\kappa_H = L_H / \mu_H\) is the condition number in Corollary 1 and \(c(\cdot)\) is defined in (35). Under the assumptions of Theorem 1, suppose \(\epsilon\) is selected such that \(\gamma(\delta + \epsilon) < 1\). Then \(\delta + \epsilon\) is the parameter for spectral sparsification and is controlled by the user. In Theorem 2, we showed that our methods transform the input Laplacian \(L\) into an approximation \(\tilde{L}\) such that \(L \approx \epsilon \tilde{L}\). The composition of the two approximation relations thus gives \(H(R) \approx \delta + \epsilon (L \otimes I_p)\), which intuitively explains why the convergence rate depends on a function of \(\delta + \epsilon\). Last, we note that while our theoretical convergence guarantees require \(\gamma(\delta + \epsilon) < 1\), our experiments (see Section VI) show that Algorithm 4 is not sensitive to the choice of \(\epsilon\) and converges under a wide range of parameter settings.

**Remark 6 (Communication efficiency of Algorithm 4):** In Algorithm 4, note that only a single call to SPARSIFIEDSCHURCOMPLEMENT (Algorithm 2) is needed, which incurs a total upload of \(O(m|C| \log |C|)\); see Remark 5. In each iteration, a single call to SPARSIFIEDLAPLACIAN SOLVER (Algorithm 3) is made, which requires a single round of upload and download. Furthermore, by Remark 5, both upload and download costs are bounded by \(O(m|C|)\). Therefore, after \(K > 0\) iterations, Algorithm 4 uses a total upload of \(O(m|C| \log |C| + mK|C|)\) and a total download of \(O(mK|C|)\). In particular, the terms that involve the number of iterations \(K\) scales linearly with the number of separators \(|C|\), which makes the algorithm very communication-efficient.

### C. Collaborative Translation Estimation

Similar to rotation averaging, we can develop a fast and communication-efficient method to solve translation estimation, which is equivalent to the Laplacian system (22) as shown in Section IV-B. Specifically, we employ our collaborative Laplacian solver (see Section V-A) in an iterative refinement framework. Let \(M_k \in \mathbb{R}^{n \times d}\) be our estimate for the translation variables at iteration \(k\) (in practice \(M_k^0\) can simply be initialized at zero). We seek a correction \(D_k\) to \(M_k\) by solving the residual system corresponding to (22):

\[
L(M_k^i + D_k) = B_i \iff LD_k = B_i - LM_k^i \triangleq E_k. \tag{37}
\]

Observing that the system on the right-hand side of (37) is another Laplacian system in \(L \equiv L(G, \tau)\), we can deploy our Laplacian solver to find an approximate solution \(D_k\). Algorithm 5 shows the pseudocode, which shares many similarities with the proposed collaborative rotation averaging method Algorithm 4. In particular, the computation of the right-hand side \(E_k\) (line 4–8) and the update step (line 11–14) are performed in a distributed fashion. The two methods also share the same communication

---

**Algorithm 5: Collaborative translation estimation.**

1: Initialize translation estimates \(M_0^i = 0_{n \times d}\).  
2: \(\tilde{S} = \text{SPARSIFIEDSCHURCOMPLEMENT}(L, \epsilon)\).  
3: for iteration \(k = 0, 1, \ldots\) do  
4: // Distributed computation of \(E_k\)  
5: Server computes \(E_k^i\) that corresponds to all separators.  
6: for each robot \(\alpha\) in parallel do  
7: Compute \(E_k^\alpha\) that corresponds to interior \(F_\alpha\).  
8: end for  
9: // Single round of communication to compute \(D_k\)  
10: Solve \(D_k = \text{SPARSIFIEDLAPLACIAN SOLVER}(L, E_k, \tilde{S})\).  
11: // Distributed updates of all translations: \(M_{k+1}^i = M_k^i + D_k\).  
12: for each robot \(\alpha\) in parallel do  
13: Update iterates by \(t_i^{k+1} = t_i^k + (D_k^\alpha)^T\) for each translation variable \(t_i\) owned by robot \(\alpha\).  
14: end for  
15: end for
complexity; see Remark 6. The following theorem states the theoretical guarantees for Algorithm 5.

**Theorem 4 (Convergence rate of Algorithm 5):** Suppose \( c(\epsilon) \) is selected such that the constant \( c(\epsilon) \) defined in (33) satisfies \( c(\epsilon) < 1 \). Let \( M^*_k \) be an optimal solution to Problem 2 and let \( M^k \) denote the solution computed by Algorithm 5 at iteration \( k \geq 1 \). It holds that
\[
\|M^k - M^*_k\|_L \leq c(\epsilon)^k \|M^*_k\|_L
\]
where \( L \equiv L(G; \tau) \).

We prove Theorem 4 in [23, Appendix IV.C]. Theorem 4 is simpler compared to its counterpart for rotation averaging (see Theorem 3). The convergence rate (38) only depends on the sparsification parameter \( \epsilon \). Furthermore, since the translation estimation problem is convex, the convergence guarantee is global and holds for any initial guess. While Theorem 4 requires \( c(\epsilon) < 1 \), our experiments show that Algorithm 5 is not sensitive to the choice of sparsification parameter \( \epsilon \) and converges under a wide range of parameter settings.

D. Extension to Outlier-Robust Optimization

So far, we have considered estimation using the standard least squares cost function, which is sensitive to outlier measurements that might arise in practice (e.g., due to incorrect loop closures in multirobot SLAM). In this section, we present an extension to outlier-robust optimization by embedding the developed solvers in the GNC framework [21, 22]. We select GNC for its good performance as reported in recent works [6, 21]. However, similar robust optimization frameworks such as iterative reweighted least squares [61] can also be used. Consider robust estimation using the truncated least squares (TLS) cost\(^4\)

\[
\min_{x \in \mathcal{X}} \sum_{(i,j) \in \mathcal{E}} \rho_{\text{TLS}}(e_{ij}(x)). \quad (39)
\]

In (39), \( x \in \mathcal{X} \) is the model to be estimated, and \( e_{ij}(x) \) is the measurement error associated with edge \((i,j) \in \mathcal{E}\) in the measurement graph. For the robust rotation extension averaging (Problem 1), we define \( x = (R_1, \ldots, R_n) \in \text{SO}(d)^n \), and \( e_{ij}(x) = \sqrt{R_{ij}/2} d(R_i, R_j) \) where \( d(\cdot, \cdot) \) is the geodesic or the chordal distance. For the robust extension of translation estimation (Problem 2), we define \( x = (t_1, \ldots, t_n) \in \mathbb{R}^{d \times n} \) and \( e_{ij}(x) = \sqrt{t_{ij}/2} \) \( | \vec{t}_j - \vec{t}_i(\cdot) | \). Notice that \( e_{ij}(x) \) is simply the square root of a single cost term in Problem 1 or Problem 2. Finally, \( \rho_{\text{TLS}}(\epsilon) \triangleq \min(\epsilon^2, \tau^2) \) denotes the TLS cost function, where \( \tau \) is a constant threshold that specifies the maximum acceptable error of inlier measurements. Intuitively, the TLS cost function achieves robustness by eliminating the impact of any outliers with error larger than \( \tau \).

To mitigate the nonconvexity introduced by robust cost functions, GNC solves (39) by optimizing a sequence of easier (i.e., less nonconvex) surrogate functions \( \rho_{\mu}^{\text{TLS}} \) that gradually converges to the original, highly nonconvex cost function \( \rho_{\text{TLS}} \). Here, \( \mu \) is the control parameter and for the TLS function, it satisfies that (i) \( \rho_{\mu}^{\text{TLS}} \) is convex for \( \mu \to 0 \), and (ii) \( \rho_{\mu}^{\text{TLS}} \) recovers \( \rho_{\text{TLS}} \) for \( \mu \to +\infty \); see [21, Example 2]. In practice, we initialize by setting \( \mu \approx 0 \), and gradually increase \( \mu \) as optimization progresses. Furthermore, leveraging the Black–Rangarajan duality [22], each surrogate problem can be formulated as follows:

\[
\min_{x \in \mathcal{X}, w_{ij}^{\text{GNC}} \in [0, 1]} \sum_{(i,j) \in \mathcal{E}} [w_{ij}^{\text{GNC}} e_{ij}(x) + \Phi_{\mu}(w_{ij}^{\text{GNC}})]. \quad (40)
\]

In (40), \( w_{ij}^{\text{GNC}} \) is a mutable weight attached to the measurement error \( e_{ij} \), and \( \Phi_{\mu} \) acts as a regularization term on the weight whose expression depends on the control parameter \( \mu \).

GNC leverages (40) by performing alternating updates on the model \( x \) and the weights \( w_{ij}^{\text{GNC}} \), while simultaneously updating the control parameter \( \mu \). Specifically, each GNC outer iteration consists of three steps:

1) **Variable Update:** optimize the surrogate problem (40) with respect to \( x \), under fixed weights \( w_{ij}^{\text{GNC}} \). Notice that this amounts to a standard weighted least squares problem

\[
\min_{x \in \mathcal{X}} \sum_{(i,j) \in \mathcal{E}} w_{ij}^{\text{GNC}} e_{ij}(x)^2. \quad (41)
\]

2) **Weight Update:** optimize the surrogate problem (40) with respect to all \( w_{ij}^{\text{GNC}} \), under fixed model \( x \). For TLS, the resulting \( w_{ij}^{\text{GNC}} \) has a closed-form solution

\[
w_{ij}^{\text{GNC}} \left\{ \begin{array}{ll}
0, & \text{if } e_{ij}^2 \leq \frac{\mu+1}{\mu+1} - \mu,
\frac{\tau}{e_{ij}} \sqrt{\mu(\mu+1) - \mu}, & \text{if } e_{ij}^2 \in \left[ \frac{\mu+1}{\mu+1} - \mu, \frac{\mu+1}{\mu+1} \right],
1, & \text{if } e_{ij}^2 \in \left[ 0, \frac{\mu+1}{\mu+1} \right],
\end{array} \right.
\quad (42)
\]

where \( e_{ij} \equiv e_{ij}(x) \) is the current measurement error.

3) **Parameter Update:** update control parameter \( \mu \) via \( \mu \leftarrow 1.4\mu \) (recommended in [21, Remark 5]), and move on to the next surrogate problem.

Initially, all measurement weights are initialized at one.

**Algorithm 6: Outlier-robust rotation averaging with GNC.**

1: Initialize control parameter \( \mu \) and measurement weights by setting \( w_{ij} = 1 \) for all measurements \((i,j) \in \mathcal{E}\).

2: while not converged do

3: **Variable update:** under fixed weights, solve the weighted rotation averaging problem by executing Algorithm 4 under the server-client architecture.

4: **Weight update:** in parallel, server computes (42) for all inter-robot measurements \( \mathcal{E}_s \), and each robot \( \alpha \) computes (42) for its local measurements \( \mathcal{E}_s \).

5: **Parameter update:** in parallel, server and all robots updates the control parameter \( \mu \).

6: end while

Next, we show that our algorithms developed in this work can be used within GNC to perform outlier-robust optimization. Algorithm 6 shows the pseudocode for robust rotation averaging (the case for translation estimation is analogous). The main observation is that, in the context of robust rotation averaging and translation estimation, the weighted least squares problems (41) solved during the variable update step have identical forms as Problems 1 and 2. The only difference is that each measurement is now discounted by the GNC weight \( w_{ij}^{\text{GNC}} \), as shown in (41).

Therefore, we can use Algorithm 4 to perform the variable update for rotation averaging (line 3), and Algorithm 5 for

---

\(^4\)Other robust cost functions, such as the Geman McClure function, can also be used in the same framework; see [21].
translation estimation. Furthermore, the weight update step can also be executed under the server–client architecture, where each robot $\alpha$ computes (42) for its local measurements $E_\alpha$, and the server handles the inter-robot measurements $E_i$; see line 4. Last, the server and all robots can in parallel perform the parameter update step by updating their local copies of the control parameter $\mu$ (line 5).

**Remark 7 (Implementation details of GNC):** We discuss several implementation details for GNC.

1) **Initialization:** Prior works (e.g., [6]) have observed that using an outlier-free initial guess when solving the variable update step is critical to ensure good performance. For multirobot SLAM, we adopt the method described in [6, Section V-B] that aligns each robot’s odometry in the global reference frame by solving a robust single pose averaging problem. Notably, this method does not require iterative communication and hence is very efficient.

2) **Known Inliers:** In many cases, a subset of measurements $E_{in} \subseteq E$ are known to be inliers. For instance, $E_{in}$ may contain robots’ odometry measurements. In our implementation, we use the standard least squares cost for $E_{in}$ and only apply GNC on the remaining measurements.

3) **Approximate Optimization:** Recall that each outer iteration of GNC invokes Algorithms 4 or 5 to perform the variable update step. Thus, when the number of outer iterations is large, the resulting optimization might become expensive in terms of both runtime and communication. However, in practice, we observe that GNC only requires a few outer iterations before the resulting estimates stabilize (see Section VI-C). This suggests that instead of running GNC to full convergence (i.e., fully classifying each measurement as either inlier or outlier), we can perform approximate optimization by limiting the number of outer iterations while still achieving robust estimation. In our experiments, we set the maximum number of GNC outer iterations to 20.

We conclude this subsection by noting that the linear convergence results (see Theorems 3 and 4) we prove in this article only hold for the outlier-free case. Extending the linear convergence to the case with outliers is challenging because GNC (and the similar method of iterative reweighted least square) is itself a heuristic. Nevertheless, our experiments demonstrate that in practice, the proposed outlier-robust extension is very effective and produces accurate solutions on real-world SLAM and SfM problems contaminated by outlier measurements.

### VI. Experimental Results

In this section, we extensively evaluate our proposed methods and demonstrate their fast convergence and communication efficiency. In addition, we show that the combination of our rotation estimation and translation estimation algorithms can be used for accurate PGO initialization. Sections VI-A and VI-B show evaluations using synthetic and benchmark datasets. Then, Sections VI-C and VI-D demonstrate outlier-robust estimation using our approach on real-world collaborative SLAM and SfM problems. Last, Section VI-E provides additional discussions on the performance of our approach in real-world problem instances. All proposed algorithms (including the GNC extension in Section V-D) are implemented in MATLAB. Some experiments use GTSAM [62] and the Theia SfM library [63] for comparison, where we run their original implementations in C++. All experiments are performed on a computer with an Intel i7-7700 K CPU and 16 GB RAM, and communication is simulated in memory in MATLAB.

**Performance Metrics:** In the experiments, we use the following metrics to evaluate algorithm performance. First, we compute the evolution of gradient norm that measures the rate of convergence. Second, to quantify communication efficiency, we record the total communication used by an algorithm. For the server–client architecture, communication is reported for both the upload and download stages. When evaluating the proposed PGO initialization method, we also compute the relative optimality gap in the cost function, defined as $(f_{\text{init}} - f_{\text{opt}}) / f_{\text{opt}}$, where $f_{\text{init}}$ and $f_{\text{opt}}$ denote the cost achieved by our initialization and the global minimizer, respectively. Last, we also report the solution distance to the global minimizer and optionally to the ground truth (the latter is only available in our synthetic experiments). Specifically, for rotation estimation, we compute the distance between our solution $\hat{R} \in SO(d)^n$ and the reference $R^{\text{ref}} \in SO(d)^n$ (either global minimizer or ground truth) using the orbit distance

$$\text{RMSE}(\hat{R}, R^{\text{ref}}) \triangleq \min_{S \in \text{SO}(d)} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left\| S \Delta \hat{R}_i - R^{\text{ref}}_i \right\|^2_F}, \quad (43)$$

Intuitively, (43) computes the root-mean-square error (RMSE) between two sets of rotations after alignment by a global rotation. The optimal alignment $S$ in (43) has a closed-form expression; see [60, Appendix C.1]. Similarly, for translations, we report the RMSE between our solution and the reference after a global alignment.

#### A. Evaluation of Estimation Accuracy and Communication Efficiency

In this section, we evaluate the estimation accuracy and communication efficiency of the proposed methods under varying problem setups and algorithm parameters. Unless otherwise mentioned, we initialize Algorithm 4 using the distributed chordal initialization approach in [9], where the number of iterations is limited to 50. Our experiments mainly consider rotation averaging problems under the chordal distance metric. We provide additional results using the geodesic distance in [23, Appendix VI].

**Impact of Spectral Sparsification on Convergence and Communication:** First, we evaluate the impact of spectral sparsification on convergence rate and communication efficiency. We start by evaluating the proposed collaborative rotation averaging solver (Algorithm 4), by simulating a 5-robot problem using the Cubicle dataset. In [23, Appendix VI], we present similar analysis for translation estimation. Recall that Algorithm 4 calls the **SPARSIFIEDSCHUR** procedure (Algorithm 2), which requires each robot $\alpha$ to transmit its sparsified matrix $S_\alpha$. Fig. 5(a) shows the number of nonzero entries in $S_\alpha$ as a function of the sparsification parameter $\epsilon$. Note that when $\epsilon = 0$, sparsification is effectively skipped and each robot transmits its exact $S_\alpha$ matrix that is potentially large and dense. In Fig. 5(a), this is reflected on robot 1 (blue curve) whose exact $S_1$ matrix has more than $2 \times 10^4$ nonzero entries and hence is expensive to transmit. However, spectral sparsification significantly reduces the density of the matrix and hence improves communication efficiency. In particular, for robot 1, applying sparsification with
shows the evolution of the estimation RMSE $n_{\epsilon H}$, where Block-Tree visualizes the accuracy as a function of total uploads $= 5750$ as a function sparsification parameter have similar slopes, which suggests $n_{1 \epsilon}$ visualizes the accuracy as a function of total dataset. (a) For each robot $\alpha$, we show the number of nonzero entries (nnz) in its sparsified matrix $\tilde{S}_\alpha$ as a function sparsification parameter $\epsilon$. (b) Evolution of Riemannian gradient norm as a function of iterations. (c) Evolution of Riemannian gradient norm as a function of total uploads. (d) Evolution of Riemannian gradient norm as a function of total downloads. (a) Sparsity of $\tilde{S}_\alpha$, (b) Gradient norm vs. iterations. (c) Gradient norm vs. uploads. (d) Gradient norm vs. downloads.

\( \epsilon = 2 \) creates a sparse $\tilde{S}_\alpha$ with 2300 nonzero entries, which is much sparser than the original $S_\alpha$.

Next, we evaluate the convergence rate and communication efficiency of Algorithm 4 with varying sparsification parameter $\epsilon$. We introduce three baseline methods for the purpose of comparison. The first baseline, called \texttt{Newton} in Fig. 5, implements the exact Newton update using domain decomposition, where each robot communicates its exact (dense) Schur complement similar to DDF-SAM [14]. In addition, we also implement two baselines that apply heuristic sparsification to \texttt{Newton}: in Block-Diagonal, each robot only transmits the diagonal blocks of its Schur complement, whereas in Block-Tree, each robot transmits both diagonal blocks and off-diagonal blocks that form a tree sparsity pattern. These two baselines are similar to the Jacobi and tree preconditioning [37], as well as the approximate summarization strategy in DDF-SAM 2.0 [16]. Fig. 5(b) shows the accuracy achieved by all methods (measured by norm of the Riemannian gradient) as a function of iterations. As expected, \texttt{Newton} achieves the best convergence speed and converges to a high-precision solution in two iterations. However, when combined with heuristic sparsifications in Block-Diagonal and Block-Tree, the resulting methods have very slow convergence. Intuitively, this result shows that a diagonal or tree sparsity pattern is not sufficient for preserving the spectrum of the original dense matrix. In contrast, our proposed method achieves fast convergence under a wide range of sparsification parameter $\epsilon$. Furthermore, by varying $\epsilon$, the proposed method provides a principled way to trade off convergence speed with communication efficiency.

Fig. 5(c) visualizes the accuracy as a function of total uploads to the server. Since both the Hessian and Laplacian matrices are symmetric, we only record the communication when uploading their upper triangular parts as sparse matrices. To convert the result to kilobyte (kB), we assume each scalar is transmitted in double precision. Our results show that the proposed method achieves the best communication efficiency under various settings of the sparsification parameter $\epsilon$. Moreover, even without sparsification (i.e., $\epsilon = 0$), the proposed method is still more communication-efficient than \texttt{Newton}. This result is due to the following reasons. First, since the Hessian matrix varies across iterations, \texttt{Newton} requires communication of the updated Hessian Schur complements at every iteration. In contrast, the proposed method works with a constant graph Laplacian, and hence only requires a one-time communication of its Schur complements; see line 2 in Algorithm 4. Second, \texttt{Newton} requires communication to form the Schur complement of the original $pn$-by-$pn$ Hessian matrix, where $n$ is the number of rotation variables and $p = \dim \text{SO}(d)$ is the intrinsic dimension of the rotation group (for the \textit{Cubicle} dataset, $n = 5750$ and $p = 3$). In contrast, the proposed method operates on the smaller $n$-by-$n$ Laplacian matrix, and the decrease in matrix size directly translates to communication reduction.

Last, Fig. 5(d) visualizes the accuracy as a function of total communication in the download stage. Notice that the evolution follows the same trend as Fig. 5(b), where the horizontal axis shows the number of iterations. This observation is expected as a result of Remark 6, which shows that the communication complexity in the download stage is $O(mK|C|)$, i.e., the total downloads grows linearly with respect to the number of iterations $K$.

\textbf{Scalability with Number of Robots:} In this experiment, we evaluate the scalability of Algorithm 4. For this purpose, we generate a large-scale synthetic rotation averaging problem with 8000 rotations arranged in a 3-D grid [see Fig. 6(a)]. With probability 0.3, we add relative measurements between nearby rotations, which are corrupted by Langevin noise with a standard deviation of 5°. Then, we divide the dataset to simulate increasing number of robots, and run Algorithm 4 with sparsification parameter $\epsilon = 0.5$ until the Riemannian gradient norm reaches $10^{-5}$. Fig. 6(b) shows the evolution of the estimation RMSE with respect to the ground truth rotations. For reference, we also show the RMSE achieved by the global minimizer to Problem 1 (denoted as “MLE” in the figure). Note that due to measurement noise, the MLE is in general different from the ground truth. The proposed method is able to achieve an RMSE similar to the MLE after a single iteration, despite the worse initialization as the number of robots increases.

Fig. 6(c) shows the evolution of gradient norm as a function of iterations. Note that all curves in Fig. 6(c) have similar slopes, which suggests that the empirical convergence rate of our method is not sensitive to the number of robots. This observation is compatible with the (local) convergence rate established in Theorem 3, which does not depend on the number of robots $m$. This property makes our method more appealing than existing fully distributed methods, whose convergence speed typically degrades as the number of robots increases (e.g., see [10, Fig. 8]). Last, Fig. 6(d) shows the evolution of total uploads as a function of iterations. As we divide the dataset to simulate more robots, both the number of inter-robot measurements and the number of separators $|C|$ increase, and thus each iteration requires more communication.
shows the results. We find that Algorithm 4 is relatively insensitive to the measurement noise, and starts to converge to suboptimal local minima as the noise level increases above 25 deg. Nevertheless, we note that the level of rotation noise encountered in practice is usually much lower, and thus we expect our algorithm to still provide effective estimation (see real-world evaluations in Sections VI-C and VI-D).

Sensitivity to Initial Guess: So far, we have used the distributed chordal initialization technique [9] to initialize Algorithm 4. In the next experiment, we test the sensitivity of our proposed method to poor initial guesses. For this purpose, we use a 9-robot simulation where each robot owns 512 rotation variables, and generate synthetic initial guesses by perturbing the global minimizer with increasing levels of Langevin noise. Using the synthetic initialization, we run Algorithm 4 with sparsification parameter $\epsilon = 0.5$ until the Riemannian gradient norm reaches $10^{-5}$ or the number of iterations exceeds 50. At each noise level, 10 random runs are performed. Fig. 7(a) shows the fraction of trials that successfully converge to the global minimizer. We observe that Algorithm 4 enjoys a large convergence basin: the success rate only begins to decrease at a large initial guess error of 35 deg. Fig. 7(b) shows the number of iterations used by Algorithm 4 to reach convergence. Our results suggest that the proposed method is not sensitive to the quality of initialization and usually requires a small number of iterations to converge.

Sensitivity to Measurement Noise: Next, we analyze the sensitivity of Algorithm 4 to increasing levels of measurement noise. The setup is similar to the previous experiment, where we use a 9-robot simulation and each robot owns 512 rotations. However, instead of varying the quality of the initial guess, we vary the noise level when generating the synthetic problem. Fig. 8 shows the results. We find that Algorithm 4 is relatively more sensitive to the measurement noise, and start to converge to suboptimal local minima as the noise level increases above 25 deg. Nevertheless, we note that the level of rotation noise

5Here, we only consider rotation noise of inlier measurements. Outlier measurements will be handled using the robust optimization framework presented in Section V-D.
and only requires a few iterations

We note

sparsifies its

and

for which

R

> 10 and

α

dataset, for which our method uses more

INTEL

Newton

Grid shows the

Rim and

to the server. This step

matrix and transmits the result

˜Rim

method. Both methods are terminated

60 deg. on the rotation averaging subproblems ex-

shows

and

on

and typically converges in a few iterations.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
In this section, we show that our approach can be used to achieve robust PGO initialization in a real-world collaborative SLAM scenario with outlier measurements. For this purpose, we collected three sets of trajectories using a Clearpath Jackal robot equipped with a front-facing RealSense D455 RGBD camera and IMU. Each trajectory covers a different area outside a building on the MIT campus, with the robot making multiple loops within the designated area. The three trajectories also overlap in a small region such that common features are observed [see Fig. 11(a)]. We run Kimera-Multi [6] to process the
We demonstrate the two-stage PGO initialization as described in Section III-D. To account for outliers, we use the GNC-based robust optimization during both rotation estimation and translation estimation stages. In our experiment, we observe that setting the TLS threshold to a smaller value of 0.5 deg for rotation estimation leads to better performance. The TLS threshold for translation estimation is set to 0.25 m. With this setting, the two-stage initialization rejects 1090 out of 1540 loop closures (71%). In [23, Appendix VI], we provide additional evaluation on the sensitivity to the TLS thresholds. Table III reports statistics and the accuracy achieved by our robust initialization for each robot. As ground truth trajectories are not available, we compare against a reference solution computed by the GNC-based robust PGO solver implemented in GTSAM [62]. While standard initialization (without GNC) has large errors, using GNC achieves robust initialization, and the final rotation and translation RMSE over all robots are 3.0 deg and 0.57 m, respectively. Fig. 11(c) demonstrates the effectiveness of our approach to achieve communication-efficient collaborative rotation and translation estimation. We report the performance of spectral sparsification on the real-world collaborative SLAM dataset. The resulting multirobot pose graph contains a 3-D pose variable for each keyframe generated by visual-inertial odometry, and each robot has a backbone of odometry measurements that are free of outliers. However, there are many outlier loop closures (both within each robot’s trajectory and between different robots), due to incorrect visual feature matching.

Based on the image IDs, we equally partition each dataset to simulate five robots and run our GNC-based rotation averaging solver, with the TLS threshold set to 5 deg. Compared to collaborative SLAM, in SfM there is a significantly larger number of inter-robot measurements. Furthermore, each robot no longer has an outlier-free odometry backbone in SfM. This means that we cannot use the approach of [6] to compute an outlier-free initial guess for the variable update step in GNC (see Remark 7). Instead, we use the initial guess from Theia that is computed using a spanning tree of the measurement graph. Table IV reports the mean estimation error. On 14 out of the 15 datasets, our GNC-based method produces accurate results with performance on par with the centralized Theia library. The only failure case, Gendarmenmarkt, is known to be a very challenging case in which the underlying 3-D scene is highly symmetric, leading to a significantly lower percentage of inlier measurements. The initial guess has a large error, which both GNC and Theia are unable to recover from. This issue could potentially be addressed with a better initialization method (e.g., using pairwise consistency maximization [66]), which we leave for future work. In summary, we conclude that on most datasets, our proposed rotation averaging solver combined with GNC is able to achieve robust rotation estimation, despite outlier measurements and the increased noise level present in internet images.

D. Evaluation on Real-World Structure-From-Motion Datasets

Last, we evaluate our method on rotation averaging problems extracted from 15 real-world SfM datasets [64]. Each dataset is a collection of many internet images taken at a particular location. We use Theia [63] to process each dataset and extract a rotation averaging problem with outliers (caused by incorrect feature matching). As ground truth is not available, we follow [63] and use 3-D reconstructions produced by the incremental SfM pipeline [65] as reference solutions. Table IV reports full dataset statistics.

We report the performance of spectral sparsification and the total communication costs of our method. For our SfM experiment, we increase the sparsification parameter to $\epsilon = 5$. In Section VI-E, we explain the reasons behind using the increased value for $\epsilon$. Table IV shows the achieved sparsity as the average ratio between the number of nonzero elements in the sparsified matrix and the input (dense) matrix. On all datasets, spectral sparsification significantly improves sparsity to as low as 40.7% on the largest Trafalgar dataset. These results, together with the total amounts of uploads and downloads, demonstrate the effectiveness of our approach to achieve communication efficiency.
TABLE IV
ROBUST ROTATION AVERAGING ON REAL-WORLD SFM DATASETS

| DATASETS          | | | Measurement Quality (%) | Mean Error (deg) | Achieved sparsity (%) | Communication (MB) |
|-------------------|---|---|--------------------------|------------------|---------------------|-------------------|
|                   | $|V|$ | $|E|$ | Inlier | Outlier | Other | Initial | No GNC | With GNC | Theta | Download | Upload |
| Montreal Notre Dame | 468 | 49705 | 81 | 4 | 15 | 4.2 | 3.3 | 1.1 | 1.1 | 55.1 | 2.09 | 1.21 |
| Ellis Island      | 241 | 19507 | 63 | 1 | 26 | 7.0 | 5.6 | 2.3 | 2.4 | 66.0 | 0.95 | 0.57 |
| NYC Library       | 355 | 17579 | 61 | 6 | 33 | 4.3 | 4.3 | 2.3 | 2.5 | 75.8 | 1.24 | 0.83 |
| Notre Dame        | 553 | 97764 | 70 | 9 | 21 | 3.5 | 4.5 | 2.4 | 2.6 | 41.8 | 2.85 | 1.54 |
| Roman Forum       | 1099 | 53989 | 74 | 3 | 23 | 16.5 | 5.1 | 2.5 | 2.5 | 68.1 | 4.43 | 2.73 |
| Alamo             | 606 | 87725 | 74 | 3 | 23 | 8.0 | 4.5 | 2.9 | 2.8 | 44.7 | 2.62 | 1.39 |
| Madrid Metropolis | 379 | 18811 | 47 | 20 | 33 | 7.7 | 8.0 | 3.4 | 3.4 | 70.3 | 1.48 | 1.08 |
| Yorkminster       | 448 | 24416 | 73 | 5 | 22 | 8.3 | 4.5 | 3.4 | 3.3 | 76.5 | 1.69 | 1.1 |
| Tower of London   | 493 | 19798 | 76 | 3 | 21 | 7.4 | 4.7 | 3.5 | 3.4 | 75.3 | 2.00 | 1.35 |
| Trafalgar         | 5433 | 680012 | 63 | 7 | 30 | 20.0 | 6.4 | 3.5 | 3.3 | 40.7 | 35.89 | 24.97 |
| Piazza del Popolo | 343 | 22342 | 82 | 4 | 14 | 5.4 | 7.8 | 3.6 | 3.5 | 70.0 | 1.29 | 0.8 |
| Piccadilly        | 2436 | 254175 | 58 | 10 | 32 | 13.9 | 14.6 | 4.9 | 5.0 | 53.1 | 14.23 | 9.79 |
| Union Square      | 930 | 25561 | 57 | 6 | 37 | 11.9 | 10.9 | 6.0 | 8.6 | 82.1 | 4.15 | 3.37 |
| Vienna Cathedral  | 900 | 96546 | 70 | 6 | 24 | 13.9 | 9.6 | 8.9 | 8.6 | 51.9 | 4.62 | 3.13 |
| Gendarmenmarkt    | 723 | 42980 | 56 | 27 | 37 | 45.0 | 40.8 | 38.1 | 38.1 | 63.8 | 4.13 | 3.03 |

Each dataset is divided to simulate 5 robots. $|V|$ and $|E|$ denote the total number of rotation variables and measurements, respectively. Using the reference solution, we quantify the difficulty of each dataset by computing the percentage of high-quality inlier measurements (measurement error < 5 deg) and gross outliers (measurement error > 45 deg). For the proposed method, we show the sparsity achieved by sparsification (lower is better) and total communication.

Fig. 12. Impact of the density of exact Schur complements on the performance of spectral sparsification. For each dataset, we select one robot and visualize the sparsity pattern of its exact Schur complement (corresponding to $S_{\alpha}$ in Algorithm 2), and the result after spectral sparsification (corresponding to $\tilde{S}_{\alpha}$ in Algorithm 2). Entries in the matrix are color-coded based on their magnitude in log scale. (a) Garage ($\epsilon = 1.5$). (b) Manhattan ($\epsilon = 1.5$). (c) Notre Dame ($\epsilon = 5.0$).

E. Discussion

We conclude our experimental evaluations by discussing the impacts of real-world problem properties on the performance of the proposed algorithms.

Effectiveness of Laplacian Approximation in the Presence of Outliers: Our rotation averaging method exploits the fact that under small measurement residuals, the Laplacian is an effective approximation of the Hessian (see Theorem 1). When there are outlier measurements, we have seen that the approximation quality degrades, leading to increased number of iterations. An example is the Rim dataset in Table I, which is contaminated by outliers. Nonetheless, we note that this issue is mitigated when using a robust optimization framework such as GNC, since outliers will be gradually discounted and eventually rejected from the measurement graph. This is shown in Fig. 11(c). During the first two GNC outer iterations, outliers have a substantial influence on the problem, causing our method (see Algorithm 4) to use more communication rounds. However, as GNC proceeds, outliers receive increasingly small weights, and our method recovers its fast convergence. In Fig. 11(c), this is shown as the slower increase in communication rounds starting from the third GNC outer iteration.

Impact of Problem Density on Sparsification Performance: As we have seen (e.g., from Table I), spectral sparsification achieves different levels of sparsity improvement on the various SLAM and SFM datasets. This is because in our method, sparsification is applied to the Schur complements that the robots form after eliminating their interior variables (see Algorithm 2). Thus, we expect the performance of sparsification to vary depending on the density of the Schur complements. To make the discussion more concrete, we identify three types of problems and Fig. 12 shows a representative sparsification result for each case. In the first case [see Fig. 12(a)], the multirobot measurement graph is extremely sparse; consequently, the resulting Schur complements are already sparse and sparsification is not necessary. In the second case, the original measurement graph is still sparse, but the robots’ Schur complements become dense due to fill-in introduced during the elimination of interior variables. For the example in Fig. 12(b), the fill-in is visualized as patches of dense entries in the exact Schur complement, and our method is highly effective at sparsifying these dense blocks. Moreover, notice that the dense fill-ins have relatively smaller magnitudes (e.g., compared to the diagonal), and thus they can be sparsified with a smaller value of the sparsification parameter $\epsilon$. In the last case, the original measurement graph is already dense and so are the resulting Schur complements [see Fig. 12(c)]. All of the SFM datasets in Table IV belong to this category because there are many images viewing a common landmark (e.g., the
Notre Dame cathedral), albeit from different locations or angles. Consequently, a relative rotation can be estimated for many image pairs, which makes the input measurement graph dense. Since there is no significant difference in the magnitudes of different matrix entries, a larger value of $\epsilon$ is needed. Similar to the second case, sparsification is highly effective at promoting sparsity in each robot’s transmitted matrix in this case.

VII. CONCLUSION

We presented fast and communication-efficient methods for solving rotation averaging and translation estimation in multirobot SLAM, SfM, and camera network localization applications. Our algorithms leverage theoretical relations between the Hessians of the optimization problems and the Laplacians of the underlying graphs. At each iteration, robots coordinate with a central server to perform approximate second-order optimization, while using spectral sparsification to achieve communication efficiency. We performed rigorous analysis of our methods and proved that they achieve (local) linear rate of convergence. Furthermore, we proposed the combination of our solvers with GNC to achieve outlier-robust estimation. Extensive experiments in real-world collaborative SLAM and SfM scenarios validate our theoretical results and demonstrate the superior convergence rate and communication efficiency of our proposed methods.

While results are promising, this work also suggests several directions for future research. First, it remains an open problem whether a similar approach can be developed for the full PGO problem. Our preliminary analysis shows that, unlike rotation averaging, the Hessian of PGO is no longer well approximated by the corresponding graph Laplacian due to the coupling between rotation and translation terms. As a result, our current approach cannot be directly applied and additional techniques need to be considered. Secondly, the proposed algorithms assume the availability of communication links that allow all robots to participate in collaborative optimization. When some robots go offline, a practical remedy is to temporarily exclude them from optimization, but this could lead to decrease in the overall accuracy. A principled extension to cope with communication failures and evaluations under more realistic scenarios (e.g., using real-world communication protocols) would be valuable. Last, extending the algorithm to leverage the incremental nature of real-world SLAM problems is another interesting direction for future research.

ACKNOWLEDGMENT

The authors gratefully acknowledge Dr. Kaveh Fathian, Parker Lusk, Dr. Kasra Khosoussi, Dr. David M. Rosen, and anonymous reviewers for helpful comments. The authors would also like to thank Prof. Pierre-Antoine Absil for insightful discussions on the linear convergence of approximate Newton methods on Riemannian manifolds.

REFERENCES

[1] P. Schmuck, T. Ziegler, M. Karner, J. Perraudin, and M. Chli, “COVINS: Visual-inertial SLAM for centralized collaboration,” in Proc. IEEE Int. Symp. Mixed Augmented Reality Adjuncts, 2021, pp. 171–176.
[2] Y. Chang et al., “LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments,” IEEE Robot. Automat. Lett., vol. 7, no. 4, pp. 9175–9182, Oct. 2022.
[3] A. Cramariuc et al., “maplab 2.0—a modular and multi-modal mapping framework,” IEEE Robot. Automat. Lett., vol. 8, no. 2, pp. 520–527, Feb. 2023.
[4] T. Cieslewski, S. Choudhary, and D. Scaramuzza, “Data-efficient decentralised visual SLAM,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 2466–2473.
[5] P.-Y. Lafolie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, “DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams,” IEEE Robot. Automat. Lett., vol. 5, no. 2, pp. 1656–1663, Apr. 2020.
[6] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, “Kinegra-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems,” IEEE Trans. Robot., vol. 38, no. 4, pp. 2022–2038, Aug. 2022.
[7] F. Delllaert and M. Kaess, “Factor graphs for robot perception,” Foundations Trends Robot., vol. 6, no. 1/2, pp. 1–139, 2017.
[8] 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.
[9] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Delllaert, “Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models,” Int. J. Robot. Res., vol. 36, no. 12, pp. 1209–1311, 2017.
[10] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, “Distributed certifiably correct pose-graph optimization,” IEEE Trans. Robot., vol. 37, no. 6, pp. 2137–2156, Dec. 2021.
[11] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, “Asynchronous and parallel distributed pose graph optimization,” IEEE Robot. Automat. Lett., vol. 5, no. 4, pp. 5819–5826, Oct. 2020.
[12] T. Fan and T. D. Murphey, “Majorization minimization methods for distributed pose graph optimization,” IEEE Trans. Robot., 2021, doi: 10.1109/TRO.2023.3324818.
[13] R. Murai, J. Ortiz, S. Saeddi, P. H. J. Kelly, and A. J. Davison, “A robot web for distributed many-device localisation,” 2022, arXiv:2202.03314.
[14] A. Cunningham, M. Faluri, and F. Delllaert, “DDF-SAM: Fully distributed SLAM using constrained factor graphs,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2010, pp. 3025–3030.
[15] A. Cunningham, K. M. Wurm, W. Burgard, and F. Delllaert, “Fully distributed scalable smoothing and mapping with robust multi-robot data association,” in Proc. IEEE Int. Conf. Robot. Automat., 2012, pp. 1093–1100.
[16] A. Cunningham, V. Indelman, and F. Delllaert, “DDF-SAM 2.0: Consistent distributed smoothing and mapping,” in Proc. IEEE Int. Conf. Robot. Automat., 2013, pp. 5220–5227.
[17] L. Carlone, R. Tron, K. Daniilidis, and F. Delllaert, “Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization,” in Proc. IEEE Int. Conf. Robot. Automat., 2015, pp. 4677–4684.
[18] S. Zhu et al., “Very large-scale global SfM by distributed motion averaging,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4568–4577.
[19] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng, “Spectral sparsification of graphs: Theory and algorithms,” Commun. ACM, vol. 56, no. 8, pp. 87–94, 2013.
[20] D. A. Spielman and N. Srivastava, “Graph sparsification by effective resistances,” SIAM J. Comput., vol. 40, pp. 1913–1926, 2011.
[21] H. Yang, P. Antonante, V. Tzounas, and L. Carlone, “Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection,” IEEE Robot. Automat. Lett., vol. 5, no. 2, pp. 1127–1134, Apr. 2020.
[22] J. M. Black and A. Rangarajan, “On the unification of line processes, outlier rejection, and robust statistics with applications in early vision,” Int. J. Comput. Vis., vol. 19, pp. 57–91, 1996.
[23] Y. Tian and J. P. How, “Spectral sparsification for communication-efficient collaborative rotation and translation estimation,” 2022, arXiv:2210.05020.
[24] Y. T. Lee, R. Peng, and D. A. Spielman, “Sparsified cholesky solvers for SDD linear systems,” 2015, arXiv:1506.08204.
[25] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, “Sparsified cholesky and multigrid solvers for connection Laplacians,” in Proc. 48th Annu. ACM Symp. Theory Comput., 2016, pp. 842–850.
[26] P.-A. Absil, R. Mahony, and R. Sepulchre, Optimization Algorithms on Matrix Manifolds. Princeton, NJ, USA: Princeton Univ. Press, 2009.
[27] N. Boumal, An Introduction to Optim. on Smooth Manifolds, New York, NY, USA: Cambridge Univ. Press, 2023.
Yulan Tian (Member, IEEE) received the B.A. degree in computer science from UC Berkeley, Berkeley, CA, USA, in 2017, and the S.M. and Ph.D. degrees in aeronautics and astronautics from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2019 and 2023, respectively.

His current research interest includes distributed perception, estimation, and optimization with applications to multiagent systems. Dr. Tian was the recipient of the 2022 King-Sun Fu Memorial Best Paper Award from IEEE TRANSACTIONS ON ROBOTICS, a 2021 Honorable Mention from IEEE TRANSACTIONS ON ROBOTICS, and a 2020 Honorable Mention from IEEE ROBOTICS AND AUTOMATION LETTERS.

[57] D. Durfee, R. Kry, J. Peebles, A. B. Rao, and S. Sachdeva, “Sampling random spanning trees faster than matrix multiplication,” in Proc. 49th Annu. ACM SIGACT Sympos. Theory Comput., 2017, pp. 730–742.

[58] N. K. Vishnoi, “Lx=b Laplacian Solvers and Their Algorithmic Applications,” Foundations Trends Theor. Comput. Sci., vol. 8, pp. 1–141, 2013.

[59] R. Peng and D. A. Spielman, “An efficient parallel solver for SDD linear systems,” in Proc. 46th Annu. ACM Sympos. Theory Comput., 2014, pp. 333–342.

[60] 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.

[61] A. Chatterjee and V. M. Govindu, “Robust relative rotation averaging,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 958–972, Apr. 2018.

[62] F. Dellaert et al., “Georgia Tech Smoothing And Mapping (GTSAM),” 2019. [Online]. Available: https://gtsam.org/

[63] C. Sweeney, H. Tobias, and T. Matthew, “Theia: A fast and scalable structure-from-motion library,” in Proc. 23rd ACM Int. Conf. Multimed., 2015. [Online]. Available: http://theia-sfm.org

[64] K. Wilson and N. Snavely, “Robust global translations with 1DSfM,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 61–75.

[65] N. Snavely, S. M. Seitz, and R. Szeliski, “Photo tourism: Exploring photo collections in 3D,” ACM Trans. Graph., pp. 835–846, 2006.

[66] 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. Automat., 2018, pp. 2916–2923.

Jonathan P. How (Fellow, IEEE) received the B.A.Sc. degree from the University of Toronto, Toronto, Canada, in 1987, and the S.M. and Ph.D. degrees in aeronautics and astronautics from 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 at MIT.

Dr. How was the recipient of several awards, which include 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.