Algorithms and Systems for Scalable Multi-Agent Geometric Estimation

by

Yulun Tian

B.A., University of California, Berkeley (2017)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

© 2023 Yulun Tian. All Rights Reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Yulun Tian
Department of Aeronautics and Astronautics
August 7, 2023

Certified by: Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor

Certified by: Ali Jadbabaie
JR East Professor of Engineering, MIT
Thesis Supervisor

Certified by: Luca Carlone
Associate Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor

Accepted by: Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Algorithms and Systems for Scalable Multi-Agent Geometric Estimation

by

Yulun Tian

Submitted to the Department of Aeronautics and Astronautics
on August 7, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Collaborative geometric estimation, which enables multiple agents to construct globally consistent geometric models of the environment (*e.g.*, maps and robot poses) from noisy local measurements, is a crucial capability for multi-agent systems. However, achieving scalable collaborative estimation in the real world is challenging. On one hand, solving the underlying geometric optimization problems is hard due to the coupling among agents and poor numerical conditioning. On the other hand, real-world communication networks impose operational constraints (*e.g.*, in the form of available bandwidth) that need to be accounted for during deployment.

This thesis develops algorithms and systems toward enabling scalable collaborative geometric estimation, with a focus on tackling the aforementioned technical challenges. The first part of this thesis considers geometric estimation under a fully distributed communication architecture, in which agents directly communicate with each other without relying on a central server. To this end, this thesis presents distributed pose graph optimization algorithms with the goals of achieving certifiable global optimality and convergence under asynchronous communication. Leveraging the developed algorithms, this thesis then develops a complete system for distributed simultaneous localization and mapping (SLAM), and demonstrates the proposed system in large-scale urban environments where up to 8 ground robots traverse a total distance close to 8 km. The second part of this thesis tackles geometric estimation under a server-client architecture, where a server coordinates communication during collaborative optimization. To this end, this thesis presents a communication-efficient solver that enables large-scale collaborative mapping with significantly reduced communication. Furthermore, specialized solvers for collaborative rotation averaging and translation estimation are developed, which exploit spectral graph theoretic methods to achieve fast convergence. These algorithmic contributions, together with open-source code and datasets, facilitate the development of scalable multi-agent perception systems in complex environments.

Thesis Supervisor: Jonathan P. How
Title: R. C. Maclaurin Professor of Aeronautics and Astronautics, MIT

Thesis Supervisor: Ali Jadbabaie
Title: JR East Professor of Engineering, MIT

Thesis Supervisor: Luca Carlone
Title: Associate Professor of Aeronautics and Astronautics, MIT

Acknowledgments

First and foremost, I would like to thank my advisor Prof. Jonathan P. How for your continuous guidance and support over the past six years. Six years ago, I joined Jon's group as a fresh college graduate who had little experience in research. Since then, you have introduced me to the exciting world of robotics research and taught me skills that have become instrumental in my work. You helped me form confidence in myself and always encouraged me to explore things I am interested in. Your professionalism and dedication will continue to motivate me regardless of what I work on in the future.

Thank you to my wonderful committee members, Prof. Luca Carlone and Prof. Ali Jadbabaie. Thank you Luca for your support and many encouragements. I have learned so much by working with you over the past three years. I am also very grateful to Ali, for your insightful comments and challenging questions that have been very inspiring and helped me better shape my ideas. Thanks to Dr. Kasra Khosoussi and Prof. David Rosen for being my thesis readers and valuable colleagues. A special thank you to Kasra, for being an amazing mentor who taught me how to do research and helped me form my research interests.

I have been incredibly fortunate to work with many talented researchers over the past six years. Many thanks to Katherine Liu, Kyel Ok, and Prof. Nicholas Roy for helping me start my research and get hands-on with hardware. Thank you, Kyel and Katherine, for the great memories of flying drones in the forest at NASA Langley research center. A big thank you to Yun Chang, for your friendship during the many field experiments and the countless hours we spent together on the Jackal robots. Thanks to Kaveh Fathian, Carlos Nieto-Granda, Alec Koppel, Amrit Singh Bedi, Parker Lusk, Miguel Calvo-Fullana, Matt Giamou, and Loc Tran for being wonderful collaborators. I have learned a lot by working with each one of you.

Thank you to other members of my ACL family. Thank you Jesus Tordesillas, Macheng Shen, Dong-Ki Kim, Michael Everett, Kris Frey, Andrew Fishberg, Jeremy Cai, Andrea Tagliabue, Lakshay Sharma, Mason Peterson, Lena Downes, Kota Kondo, Nick Rober ... for being great labmates. A special thank you to our

amazing administrative assistant, Bryt Bradley, for your prompt help with each of my requests and questions over the past six years.

Thank you, Jinghuan Jiang, for being an amazing and trustworthy friend since high school. Thank you so much, Siyi Hu. I cannot imagine how I could otherwise navigate through the challenges of the past few years without your love and support. Lastly, I want to thank my parents. I would never have had the chance to pursue my dreams without their unconditional love all along the way.

This work was supported by ARL DCIST under Cooperative Agreement Number W911NF-17-2-0181, NASA Convergent Aeronautics Solutions project Design Environment for Novel Vertical Lift Vehicles (DELIVER), and ONR under BRC Award N000141712072.

Contents

List of Figures

List of Tables

Chapter 1

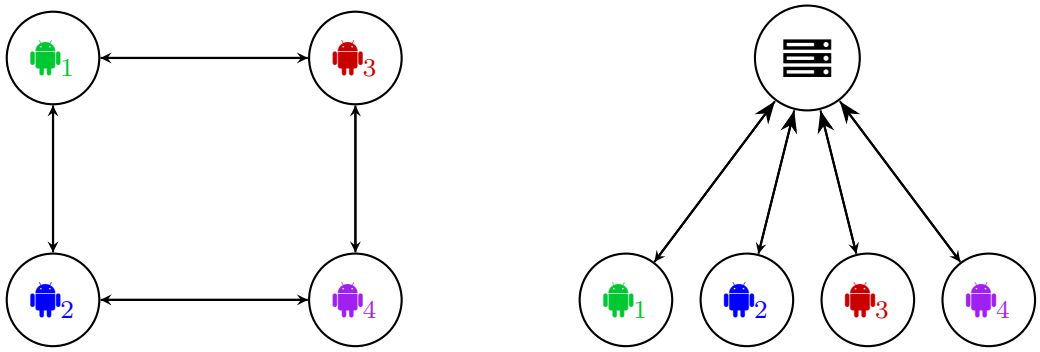
Introduction

1.1 Motivation

The past two decades have seen significant advances in single-robot spatial perception, which enables a robot to perform real-time simultaneous localization and mapping (SLAM) and semantic scene understanding for increasingly complex downstream tasks. However, many emerging applications such as disaster response, planetary exploration, and multi-user mixed reality require deploying and coordinating multiple robots and/or smart devices at once. At the forefront of these tasks is the crucial capability of **collaborative geometric estimation**, in which agents build globally consistent geometric models of the environment (*e.g.*, robot trajectories, object poses, and 3D maps) that constitute the basis of higher-level inference and planning. Such "back-end estimation" is typically accomplished by solving a large-scale and non-convex geometric optimization problem that fuses noisy observations collected by individual agents. However, efficiently solving this optimization problem is challenging due to the coupling between individual agents' estimation problems as well as various operational constraints imposed by real-world computation and communication hardware.

The Collaborative Optimization Paradigm. Existing multi-agent systems often offload the aforementioned global optimization to a powerful central server or base station [1–5]. Such a centralized paradigm offers a number of advantages, includ-

ing the ease of data management and the availability of off-the-shelf centralized solvers [6–8]. However, this approach also faces a critical scalability issue since the central server ultimately becomes a computational bottleneck as more agents participate in the collaborative estimation. In this thesis, we investigate an alternative, *collaborative* paradigm in which agents jointly solve the underlying optimization problem leveraging the availability of communication and local computation. By distributing the underlying computation and memory requirements to the entire team of agents, this framework offers significantly better scalability compared to centralized computation. Furthermore, this framework effectively mitigates privacy concerns associated with centralized computation, by avoiding sending raw data that could reveal sensitive information (*e.g.*, appearance or location data of different users).



(a) Fully distributed (b) Server-client

Figure 1-1: Communication architectures considered in this thesis.

Communication Architectures. In the collaborative optimization paradigm, communication is an integral part through which information propagates among agents, ultimately leading to convergent and globally consistent estimation. In practice, depending on the nature of the application, communication can be implemented under different architectures. In this thesis, we consider two prominent architectures that together cover a wide range of applications. In the first one, henceforth referred to as the *fully distributed architecture* (Figure 1-1a), agents achieve collaborative optimization by passing messages among themselves without the presence of a central server. The decentralized nature of this architecture makes it resilient against single point of failure. Furthermore, its flexibility makes this architecture suitable for various

types of multi-robot missions, including those in extreme environments (*e.g.* disaster response and planetary exploration) where communication is severely constrained.

In addition to the fully distributed architecture, this thesis also investigates the *server-client architecture* (Figure 1-1b), in which a central server coordinates communication among different agents (clients). Despite the use of a server, the underlying computation is still distributed across all agents to achieve scalability and protect user privacy. In related literature, this architecture is also referred to as the *federated* architecture and has received surging interest due to the recent success of federated learning (*e.g.*, [9–12]). This architecture is suitable in scenarios where a central server is readily available, *e.g.*, as a result of prior mission planning or pre-existing communication infrastructures. For example, in future smart cities, multiple devices (*e.g.*, smartphones) could contribute updates to a shared map by communicating with a remote server. In this thesis, we show that the server-client architecture brings two benefits compared to the fully distributed model: (i) it enables collaborative estimation at a higher level of granularity, and (ii) it leads to faster optimization (in the sense of significantly smaller number of iterations) by leveraging global information accessible by the server.

Technical Challenges. Despite the relative maturity of single-robot geometric estimation, achieving a similar level of accuracy and reliability in large-scale, distributed robot teams faces many technical challenges. One group of challenges arises from the underlying optimization problems. First, most problems of practical interest are *non-convex* due to geometric constraints on the search space (*e.g.*, estimating robot orientations on the group of rotations). To handle this issue, *certifiable* algorithms [13–16] that are capable of verifying global optimality of candidate solutions (or otherwise declaring failure of verification) are desirable, but are unexplored in the multi-agent setting prior to this thesis. Second, many real-world problem instances are *ill-conditioned*, which could slow down numerical optimization or even cause erratic behaviors. In our applications, poor conditioning is usually a result of the poor connectivity of the underlying measurement graph, which characterizes the couplings among the individual geometric states to be estimated. In some cases, poor condi-

tioning also arises as a result of the strong nonlinearity of the measurement model, *e.g.*, when working with projective measurements from monocular cameras. Furthermore, larger problems tend to have worse conditioning, which makes scaling to more agents or longer operation time challenging.

The second group of challenges arises from operational constraints imposed by real-world communication networks. For instance, in wireless ad hoc networks, connections among robots are usually opportunistic and subject to *high latency*. To address this issue, algorithms that are designed to cope with communication delays are highly desirable. In addition, communication constraints may also be manifested in the form of *limited bandwidth*. This issue is especially relevant when one wishes to transmit large-scale geometric models such as large 3D maps. In general, transmitting large models under limited bandwidth may result in long delays, especially if such communication needs to be repeated, *e.g.*, during iterative optimization.

1.2 Thesis Overview

This thesis develops *algorithms* and *systems* toward scalable collaborative geometric estimation, with an emphasis on addressing the technical challenges outlined in the previous section. First, we give a high-level overview of the technical approaches adopted in this thesis, before presenting the main contributions in Section 1.2.1.

From the algorithmic side, we propose collaborative optimization methods with theoretical guarantees on convergence and (local or global) optimality. Compared to the plethora of existing works on distributed optimization, the proposed approaches exploit unique properties of multi-view geometric estimation to obtain better performance. First, as the basis of all developed algorithms, we leverage the framework of *Riemannian optimization* [17, 18], which allows us to express the problems of interest as unconstrained optimization problems on certain smooth matrix manifolds. The Riemannian approach brings computational benefits (*e.g.*, by eliminating parametrization singularities caused by heuristic formulations) and also paves the way for rigorous theoretical analysis. Second, the developed algorithms exploit the

intimate connections between the optimization problems and their underlying *graph representations*, *e.g.*, pose graphs [14] and generic factor graphs [19]. Specifically, Chapters 4 and 5 leverage graph sparsity to develop distributed algorithms with very efficient communication at each iteration. Furthermore, Chapter 8 shows that graph connectivity (as captured by the graph Laplacian matrix) contains rich information that enables fast convergence in collaborative rotation averaging and translation estimation. Lastly, for an important special case of geometric estimation known as pose graph optimization, this thesis leverages recent results on its *convex relaxation* to achieve certifiably correct distributed optimization, which extends similar performance guarantees from the single-agent [14, 20] to the multi-agent domain.

From the systems side, this thesis integrates the proposed algorithms to enable multi-robot SLAM with real-world demonstrations. To this end, we focus on the development of a fully distributed metric-semantic SLAM system. The proposed system consists of a front-end for distributed loop closure based on sparse visual features, a back-end for distributed trajectory estimation based on the distributed optimization algorithm developed in this thesis, and a local mapping module for 3D dense metric-semantic mapping. Furthermore, we conduct extensive, large-scale field experiments to validate the proposed system and test its scalability.

1.2.1 Contributions

The contributions of this thesis are organized in two parts. The first part presents algorithms and systems for collaborative geometric estimation under the fully distributed architecture (Figure 1-1a). The specific contributions are:

- **Distributed Pose Graph Optimization** [21, 22]. This thesis develops distributed pose graph optimization (PGO) algorithms that serve as the estimation back-end of modern collaborative SLAM (CSLAM) systems. Chapter 4 develops a distributed PGO algorithm with certifiable global optimality. The proposed method is based upon a sparse semidefinite relaxation that provably recovers globally optimal PGO solutions under moderate measurement noise, and is fur-

thermore amenable to distributed optimization using the low-rank Riemannian Staircase framework. Chapter 5 further adapts the distributed local search algorithm to operate under asynchronous communication, thereby offering resiliency against communication delays. We prove first-order convergence and establish sublinear convergence rates for the proposed distributed local search methods. Extensive numerical evaluations on synthetic and real-world datasets demonstrate the superior performance of the proposed algorithms.

- **Distributed Metric-Semantic SLAM System** [23–25]. Leveraging the distributed PGO algorithms, Chapter 6 presents a robust and fully distributed system for multi-robot metric-semantic SLAM. The proposed system, called Kimera-Multi, is able to accurately estimate robot trajectories and dense 3D metric-semantic meshes in a common reference frame. We perform live experiments and evaluate Kimera-Multi on the resulting large-scale datasets that include up to 8 robots traversing long distances (up to 8 km) in challenging urban environments. The experiments demonstrate the resilience of Kimera-Multi under different communication scenarios, and provide a quantitative comparison with a centralized baseline system. Both the implementations and datasets are made publicly available to facilitate future research.[1]

The second part of the thesis focuses on collaborative geometric estimation under the server-client architecture (Figure 1-1b). The specific contributions are:

- **Collaborative Optimization with Event-Triggered Communication** [26]. Chapter 7 develops a communication-efficient solver for collaborative geometric estimation. The proposed method allows agents to cooperatively reconstruct a shared geometric model on the central server by fusing individual observations, but *without* the need to transmit potentially sensitive information about the agents themselves (such as their locations). Furthermore, to alleviate the burden of communication during iterative optimization, the proposed method

[1] Code is available at https://github.com/MIT-SPARK/Kimera-Multi, and datasets are available at https://github.com/MIT-SPARK/Kimera-Multi-Data.

incorporates a set of *communication triggering conditions* that enable agents to selectively upload a targeted subset of local information that is useful to global optimization. We establish first-order convergence for the proposed method. Numerical results show the proposed algorithm achieves significant communication reduction on large-scale bundle adjustment problems.

- **Collaborative Rotation Averaging and Translation Estimation with Spectral Sparsification** [27]. Chapter 8 designs specialized solvers with fast convergence for rotation averaging, translation estimation, and two-stage pose graph initialization. The proposed methods are based on theoretical relations between the Riemannian Hessians and the Laplacians of suitably weighted graphs. In the proposed methods, robots coordinate with the central server to perform approximate second-order optimization by approximately solving a Laplacian system at each iteration of Riemannian optimization. Furthermore, the proposed algorithms permit robots to employ *spectral sparsification* to sparsify intermediate dense matrices before communication, and hence provide a mechanism to trade off accuracy with communication efficiency with provable guarantees. We prove (local) *linear* rate of convergence for the proposed methods. Furthermore, the developed methods are combined with graduated non-convexity [28] to achieve outlier-robust estimation. Extensive experiments on real-world SLAM and Structure-from-Motion (SfM) scenarios demonstrate the superior convergence rate and communication efficiency of the proposed methods.

1.2.2 Related Publications

The technical contents of this thesis are based on the following publications.

- **Y. Tian**, K. Khosoussi, D. M. Rosen and J. P. How, "Distributed Certifiably Correct Pose-Graph Optimization," IEEE Transactions on Robotics, 2021. **Honorable Mention for King-Sun Fu Memorial Best Paper Award, 2021.**

- **Y. Tian**, A. Koppel, A. S. Bedi and J. P. How, "Asynchronous and Parallel Distributed Pose Graph Optimization," IEEE Robotics and Automation Letters, 2020. **Honorable Mention for Best Paper Award, 2020.**

- Y. Chang, **Y. Tian**, J. P. How and L. Carlone, "Kimera-Multi: a System for Distributed Multi-Robot Metric-Semantic Simultaneous Localization and Mapping," IEEE International Conference on Robotics and Automation (ICRA), 2021.

- **Y. Tian**, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How and L. Carlone, "Kimera-Multi: Robust, Distributed, Dense Metric-Semantic SLAM for Multi-Robot Systems," IEEE Transactions on Robotics, 2022. **King-Sun Fu Memorial Best Paper Award, 2022.**

- **Y. Tian**, Y. Chang, L. Quang, A. Schang, C. Nieto-Granda, J. P. How and L. Carlone, "Resilient and Distributed Multi-Robot Visual SLAM: Datasets, Experiments, and Lessons Learned," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023 (To Appear).

- **Y. Tian**, A. S. Bedi, A. Koppel, M. Calvo-Fullana, D. M. Rosen and J. P. How, "Distributed Riemannian Optimization with Lazy Communication for Collaborative Geometric Estimation," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.

- **Y. Tian** and J. P. How, "Spectral Sparsification for Communication-Efficient Collaborative Rotation and Translation Estimation," Under Review, 2022, https://arxiv.org/abs/2210.05020.

1.2.3 Thesis Organization

Chapter 2 presents mathematical preliminaries and formally introduces the collaborative geometric estimation problems considered in this thesis. Chapter 3 provides a summary of related works. Chapters 4 to 6 present contributions under the fully distributed communication architecture (Figure 1-1a). In particular, Chapter 4 develops

a certifiably correct distributed PGO solver. Chapter 5 extends the distributed local search algorithm to operate under asynchronous communication. Chapter 6 develops a robust and fully distributed SLAM system, Kimera-Multi, and presents evaluation results obtained from large-scale field experiments. Chapters 7 and 8 present contributions under the server-client architecture (Figure 1-1b). Specifically, Chapter 7 develops a communication-efficient collaborative solver based on event-triggered communication. Chapter 8 develops specialized solvers with fast convergence for collaborative rotation averaging and translation estimation based on spectral sparsification. Finally, Chapter 9 presents the conclusion and discusses future work.

Chapter 2

Background

This chapter presents mathematical preliminaries and formulations for understanding the collaborative geometric estimation problems considered in this thesis. Section 2.1 first introduces basic notations and concepts. Section 2.2 then formally defines the geometric estimation problems.

2.1 Mathematical Preliminaries

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

For a symmetric matrix A, $A \succeq 0$ means that A is positive semidefinite. Furthermore, $\mathcal{S}^n$ and $\mathcal{S}^n_+$ denote the set of n-by-n symmetric and symmetric positive semidefinite matrices, respectively. We use $\otimes$ to denote the Kronecker product. For a positive integer n, $1_n \in \mathbb{R}^n$ denotes the vectors of all ones, and $I_n \in \mathbb{R}^{n\times n}$ and $I_{n\times n} \in \mathbb{R}^{n\times n}$ denote 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. $A^\dagger$ denotes the Moore-Penrose inverse of A, which coincides with the inverse A^{-1} when A is invertible. When $A \in \mathcal{S}^n$, $\lambda_1(A), \ldots, \lambda_n(A)$ denote the real eigenvalues of A sorted in increasing

order. When $A \in \mathcal{S}_+^n$, we also define $\|X\|_A \triangleq \sqrt{\mathrm{tr}(X^\top A X)}$ where X is of compatible dimensions. $\mathrm{Proj}_{\mathcal{S}}$ denotes the orthogonal projection operator onto a given set $\mathcal{S}$ with respect to the Frobenius norm.

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 $w : \mathcal{E} \to \mathbb{R}_{>0}$ 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* $L(G; w) \in \mathcal{S}_+^n$ is defined as,

$$L(G;w)_{ij} = \begin{cases} \sum_{k \in \mathrm{Nbr}(i)} w_{ik}, & \text{if } i = j, \\ -w_{ij}, & \text{if } i \neq j,\ (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

In (2.1), $\mathrm{Nbr}(i) \subseteq \mathcal{V}$ 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.

Optimization on Matrix Manifolds

The reader is referred to [17, 18] for a comprehensive review of optimization on matrix manifolds. In general, we use $\mathcal{M}$ to denote a smooth matrix manifold embedded in an ambient Euclidean space. For integer $n > 1$, $\mathcal{M}^n$ denotes the product manifold formed by n copies of $\mathcal{M}$. The tangent space at $x \in \mathcal{M}$ is denoted as $T_x\mathcal{M}$, or simply as T_x when the manifold $\mathcal{M}$ is clear from context. Informally, $T_x\mathcal{M}$ contains all possible directions of change at x while staying on $\mathcal{M}$. When $\mathcal{M}$ is a matrix manifold, $T_x\mathcal{M}$ can be identified with a linear subspace of the ambient Euclidean space. 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 $\|\eta\|_x = \sqrt{\langle \eta, \eta \rangle_x}$. By default, we define the inner product by treating tangent vectors as elements of the ambient (Euclidean) space and inheriting the standard inner product from the ambient space, *i.e.*, $\langle \eta, \xi \rangle_x \triangleq \mathrm{tr}(\eta^\top \xi)$. An exception is Chapter 8, in which we use a different definition of the inner product for the rotation group; see the end of Section 8.1 for details. For the sake of brevity, we drop the subscript x from our notations $\langle \cdot, \cdot \rangle_x$ and $\|\cdot\|_x$ it will be clear from context.

A tangent vector can be mapped back to the manifold through a retraction $\mathrm{Retr}_x : T_x\mathcal{M} \to \mathcal{M}$, which is a smooth mapping that preserves the first-order structure of the manifold [18, Chapter 3.6]. The exponential map $\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M}$ is a particular retraction that produces geodesic curves on the manifold [18, Chapter 10.2]. The injectivity radius $\mathrm{inj}(x)$ is a positive constant such that Exp_x is a diffeomorphism when restricted to the domain $U = \{\eta \in T_x\mathcal{M} : \|\eta\| < \mathrm{inj}(x)\}$. In this case, we define the logarithm map to be $\mathrm{Log}_x \triangleq \mathrm{Exp}_x^{-1}$. Unless otherwise mentioned, we use $\mathbf{d}(x, y)$ to denote the geodesic distance between two points $x, y \in \mathcal{M}$ induced by the Riemannian metric. In addition, it holds that $\mathbf{d}(x, y) = \|v\|$ where $v = \mathrm{Log}_x(y)$; see [18, Proposition 10.22].

For a smooth real-valued function defined on a matrix manifold $f : \mathcal{M} \to \mathbb{R}$, we use $\nabla f(x)$ and $\mathrm{grad}\, f(x)$ to denote the Euclidean and Riemannian gradients of f at $x \in \mathcal{M}$. Under the inner product inherited from the ambient Euclidean space, the Riemannian gradient is given by the orthogonal projection of the Euclidean gradient onto the tangent space:

$$\mathrm{grad}\, f(x) = \mathrm{Proj}_{T_x\mathcal{M}}(\nabla f(x)). \tag{2.2}$$

We call $x^\star \in \mathcal{M}$ a *first-order critical point* if $\mathrm{grad}\, f(x^\star) = 0$. The Riemannian Hessian is a linear mapping on the tangent space which captures the directional derivative of

the Riemannian gradient:

$$\begin{aligned} \operatorname{Hess} f(x) : T_x\mathcal{M} &\to T_x\mathcal{M}, \\ \eta &\mapsto \operatorname{Proj}_{T_x\mathcal{M}}(\mathrm{D}\operatorname{grad} f(x)[\eta]). \end{aligned} \tag{2.3}$$

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

Details of Specific Matrix Manifolds

In the following, we discuss important matrix manifolds used throughout this thesis.

The special orthogonal group $\mathrm{SO}(d)$**.** The special orthogonal group (*i.e.*, the group of rotations) is defined as,

$$\mathrm{SO}(d) \triangleq \{R \in \mathbb{R}^{d\times d} : R^\top R = I_d,\ \det(R) = 1\}. \tag{2.4}$$

We exclusively work with 2D and 3D rotations, *i.e.*, $d \in \{2, 3\}$. The tangent space at R is given by $T_R\,\mathrm{SO}(d) = \{RV : V \in \mathrm{so}(d)\} \subset \mathbb{R}^{d\times d}$, where $\mathrm{so}(d)$ is the space of d-by-d skew-symmetric matrices. For computational benefits, we also define a basis for $T_R\,\mathrm{SO}(d)$ such that each tangent vector $\eta \in T_R\,\mathrm{SO}(d)$ is identified with a Euclidean vector $v \in \mathbb{R}^p$, where $p = \dim \mathrm{SO}(d) = d(d-1)/2$ is the dimension of the tangent space. For 3D rotations, we define $v \in \mathbb{R}^3$ such that,

$$\eta = R\,[v]_\times = R \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{2.5}$$

For 2D rotations, we define $v \in \mathbb{R}$ in an analogous way,

$$\eta = R\,[v]_\times = R \begin{bmatrix} 0 & -v \\ v & 0 \end{bmatrix}. \tag{2.6}$$

In (2.5) and (2.6), we have overloaded the notation $[\cdot]_\times$ to map the input scalar or vector to the corresponding skew-symmetric matrix in so(2) or so(3). In addition, we define the function $\mathrm{Exp} : \mathbb{R}^p \to \mathrm{SO}(d)$ as,

$$\mathrm{Exp}(v) \triangleq \exp([v]_\times), \tag{2.7}$$

where $\exp(\cdot)$ denotes the conventional matrix exponential. Note that $\mathrm{Exp} : \mathbb{R}^p \to \mathrm{SO}(d)$ should not be confused with the exponential mapping on Riemannian manifolds $\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M}$ discussed in the previous section, although the two are closely related in the case of rotations. Specifically, at a point $R \in \mathrm{SO}(d)$ where $d \in \{2, 3\}$, the exponential map can be written as $\mathrm{Exp}_R(\eta) = R\,\mathrm{Exp}(v)$. Lastly, we also denote Log as the inverse of Exp in (2.7).

The special Euclidean group $\mathrm{SE}(d)$**.** The special Euclidean group (*i.e.*, the group of rigid body transformations) is defined as,

$$\mathrm{SE}(d) \triangleq \{(R, t) : R \in \mathrm{SO}(d),\ t \in \mathbb{R}^d\}. \tag{2.8}$$

While $\mathrm{SE}(d)$ is itself a matrix Lie group, this representation is rarely used in this thesis. This is because for the purpose of performing optimization, it suffices to treat $\mathrm{SE}(d)$ as the product manifold $\mathrm{SO}(d) \times \mathbb{R}^d$ as in prior works such as [14, 29]. The Riemannian geometry, including the definitions of tangent vectors and inner products, is readily available from the product structure.

The Stiefel manifold $\mathrm{St}(d, r)$**.** The Stiefel manifold $\mathrm{St}(d, r)$ (with $d \leq r$) is defined as,

$$\mathrm{St}(d, r) \triangleq \{Y \in \mathbb{R}^{r\times d} \mid Y^\top Y = I_d\}. \tag{2.9}$$

The dimension of this manifold is given by $\dim(\mathrm{St}(d, r)) = dr - d(d + 1)/2$. Note that for $d = r$, the Stiefel manifold becomes the orthogonal group $\mathrm{O}(d)$. Given a matrix $A \in \mathbb{R}^{r\times d}$, if $A = U\Sigma V^\top$ is the singular value decomposition (SVD) of A, the

projection of A onto $\mathrm{St}(d, r)$ can be obtained as:

$$\mathrm{Proj}_{\mathrm{St}(d,r)}(A) = UV^\top. \tag{2.10}$$

2.2 Collaborative Geometric Estimation

This section formally presents the collaborative geometric estimation problems studied in this thesis. For each problem, we first present its optimization formulation derived from maximum likelihood estimation under a suitable measurement model. Then, we describe its realization in the setting of multi-agent collaborative estimation. As we will see, these problems differ in the underlying geometric states one wishes to estimate and the corresponding measurement models. However, they also share two important features. First, all problems are instances of *unconstrained* Riemannian optimization problems defined on certain smooth matrix manifolds. Second, all problems admit *sparse*, *graph-based* representations. Both properties are important and are leveraged in this thesis to develop efficient collaborative optimization algorithms.

2.2.1 Rotation Averaging

In rotation averaging, we are concerned with estimating n rotation variables given noisy relative measurements between pairs of rotations. Rotation averaging is a fundamental problem in robotics and computer vision [30]. In SLAM (*e.g.*, [31]) and SfM (*e.g.*, [32]), rotation averaging appears as a key subproblem toward estimating robot trajectories and/or building large-scale 3D maps. In distributed camera networks (*e.g.*, [33]), rotation averaging is also used to estimate the orientations of spatially distributed cameras with overlapping fields of view. We model rotation averaging using an undirected *measurement graph* $G = (\mathcal{V}, \mathcal{E})$. Each vertex $i \in \mathcal{V} = [n]$ corresponds to a rotation variable $R_i \in \mathrm{SO}(d)$ to be estimated. Each edge $(i, j) \in \mathcal{E}$ corresponds to a noisy relative measurement of the form,

$$\widetilde{R}_{ij} = \underline{R}_i^\top \underline{R}_j R_{ij}^{\mathrm{err}}, \tag{2.11}$$

where $\underline{R}_i, \underline{R}_j \in \mathrm{SO}(d)$ are the latent (ground truth) rotations and $R_{ij}^{\mathrm{err}} \in \mathrm{SO}(d)$ is the independent measurement noise. We consider two probabilistic generative models for R_{ij}^{err}, both of which can be viewed as generalizations of the standard Gaussian distribution to the rotation group. The first model assumes that R_{ij}^{err} is distributed according to the isotropic Langevin distribution [14, Appendix A]. In general, the Langevin distribution with mode $M \in \mathrm{SO}(d)$ and concentration parameter $\kappa > 0$ has the following probability density function with respect to the Haar measure on $\mathrm{SO}(d)$,

$$p(R; M, \kappa) = \frac{1}{c_d(\kappa)} e^{\kappa \operatorname{tr}(M^\top R)}. \tag{2.12}$$

In (2.12), $c_d(\kappa)$ is the normalization constant. In our first noise model, we assume that each R_{ij}^{err} is independently sampled from a Langevin distribution with mode $M = I_d$ and concentration parameter $\kappa_{ij} > 0$, *i.e.*,

$$R_{ij}^{\mathrm{err}} \sim \mathrm{Langevin}(I_d,\ \kappa_{ij}). \tag{2.13}$$

Under the generative model (2.13), maximum likelihood estimation amounts to solving the following non-convex optimization problem,

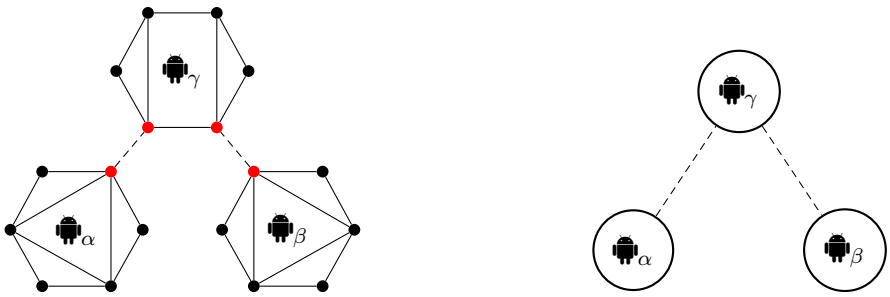
$$\underset{R=(R_1,\ldots,R_n)\in \mathrm{SO}(d)^n}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \left\| R_i \widetilde{R}_{ij} - R_j \right\|_F^2. \tag{2.14}$$

In the second noise model, we assume that R_{ij}^{err} is independently sampled from a wrapped Gaussian distribution with zero mean and covariance $\Sigma_{ij} = \kappa_{ij}^{-1} I_p$,

$$R_{ij}^{\mathrm{err}} = \mathrm{Exp}(v_{ij}^{\mathrm{err}}), \quad v_{ij}^{\mathrm{err}} \sim \mathcal{N}(0, \kappa_{ij}^{-1} I_p). \tag{2.15}$$

When Σ_{ij} is small (equivalently, when the precision κ_{ij} is large), maximum likelihood estimation under (2.15) is well approximated by the following optimization program.

$$\underset{R=(R_1,\ldots,R_n)\in \mathrm{SO}(d)^n}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \left\| \mathrm{Log}(\widetilde{R}_{ij}^\top R_i^\top R_j) \right\|^2. \tag{2.16}$$



(a) Example measurement graph (b) Robot-level dependency graph

Figure 2-1: (a) Example multi-robot measurement graph for rotation averaging or pose graph optimization (PGO). Each robot has multiple rotation or pose variables (vertices) that are connected by intra-robot and inter-robot measurements. We refer to vertices that have inter-robot measurements (dashed edges) as *separators* (marked in red), and all other vertices as *interior* vertices (marked in black). (b) Robot-level dependency graph corresponding to the measurement graph in (a). Each vertex corresponds to a robot, and two robots are adjacent if and only if there exists at least one inter-robot measurement between the corresponding robots.

Importantly, problems (2.14) and (2.16) can be combined into the following unified description, by noting that both minimize the sum of squared measurement residuals, where the each residual is evaluated using either the chordal distance or the geodesic distance.

Problem 2.1 (Rotation Averaging)**.**

$$\underset{R=(R_1,\ldots,R_n)\in \mathrm{SO}(d)^n}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \kappa_{ij}\, \mathbf{d}(R_i\widetilde{R}_{ij}, R_j)^2. \tag{2.17}$$

For each edge $(i,j) \in \mathcal{E}$, $\kappa_{ij} > 0$ is the corresponding measurement weight. The function $\mathbf{d}(\cdot,\cdot)$ refers to either the chordal distance (2.18a) or the geodesic (2.18b) distance on $\mathrm{SO}(d)$,

$$\mathbf{d}(R_i\widetilde{R}_{ij}, R_j) = \begin{cases} \mathbf{d}_{\mathrm{chr}}(R_i\widetilde{R}_{ij}, R_j) \triangleq \left\| R_i\widetilde{R}_{ij} - R_j \right\|_F. & \text{(2.18a)} \\ \mathbf{d}_{\angle}(R_i\widetilde{R}_{ij}, R_j) \triangleq \left\| \mathrm{Log}(\widetilde{R}_{ij}^\top R_i^\top R_j) \right\|. & \text{(2.18b)} \end{cases}$$

Collaborative Rotation Averaging. In Chapter 8, we study multi-robot collaborative rotation averaging. Each robot owns a subset of all rotation variables and

only knows about measurements involving its own variables. As a result, robots must collaborate to solve the overall rotation averaging problem. From the graph perspective, the entire measurement graph is partitioned based on multiple disjoint subsets of vertices (*i.e.*, rotation variables), where each subset corresponds to the variables owned by a different robot. See Figure 2-1a for an example measurement graph. The existence of inter-robot measurements (*i.e.*, edges) creates coupling among individual robots' estimation problems, which can be visualized at the level of robots as a dependency graph. Figure 2-1b shows the robot-level dependency graph that corresponds to the example in Figure 2-1a.

Definition 2.1 (Separator and interior vertices)**.** In a measurement graph, we identify a special set of vertices called the *separators*. Intuitively, the separators "separate" the remaining graph into multiple connected components, where each component contains the *interior* variables of each robot. In collaborative rotation averaging or pose graph optimization (the latter will be introduced in the next subsection), the separators correspond to the set of vertices that share inter-robot measurements. For example, in Figure 2-1a, the separators are marked in red while interior vertices are marked in black. In Chapters 4 and 5, we also refer to separators and interior vertices as public and private variables, respectively.

2.2.2 Pose Graph Optimization (PGO)

Pose graph optimization (PGO) is the crucial backbone of state-of-the-art single-robot SLAM (*e.g.*, [34–37]) and multi-robot SLAM (*e.g.*, [3, 4, 24, 38]) systems. Similar to rotation averaging, PGO can be represented using a measurement graph $G = (\mathcal{V}, \mathcal{E})$. The main difference is that each vertex $i \in \mathcal{V} = [n]$ is now augmented to represent a full pose variable $T_i \in \mathrm{SE}(d)$ to be estimated. Correspondingly, each edge $(i, j) \in \mathcal{E}$ represents a noisy relative pose measurement $\widetilde{T}_{ij} = (\widetilde{R}_{ij}, \widetilde{t}_{ij}) \in \mathrm{SE}(d)$ between poses T_i and T_j. In this thesis, we adopt the following generative noise model,

$$\widetilde{R}_{ij} = \underline{R}_i^\top \underline{R}_j R_{ij}^{\mathrm{err}}, \quad R_{ij}^{\mathrm{err}} \sim \mathrm{Langevin}(I_d,\ \kappa_{ij}). \tag{2.19a}$$

$$\widetilde{t}_{ij} = \underline{R}_i^\top (\underline{t}_j - \underline{t}_i) + t_{ij}^{\text{err}}, \quad t_{ij}^{\text{err}} \sim \mathcal{N}(0, \tau_{ij}^{-1} I_d). \tag{2.19b}$$

Above, $\underline{R}_i, \underline{R}_j \in \mathrm{SO}(d)$ and $\underline{t}_i, \underline{t}_j \in \mathbb{R}^d$ denote the latent (ground truth) rotations and translation vectors of pose i and j. Under the noise model (2.19a)-(2.19b), it can be shown that a maximum likelihood estimate is obtained as a minimizer of the following non-convex optimization problem [14]:

Problem 2.2 (Pose Graph Optimization (PGO))**.**

$$\underset{\substack{R=(R_1,\ldots,R_n)\in\mathrm{SO}(d)^n,\\ t=(t_1,\ldots,t_n)\in\mathbb{R}^{d\times n}}}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \left\| R_j - R_i \widetilde{R}_{ij} \right\|_F^2 + \tau_{ij} \left\| t_j - t_i - R_i \widetilde{t}_{ij} \right\|_2^2. \tag{2.20}$$

Collaborative PGO. Chapters 4 and 5 study collaborative PGO, in which multiple robots collaboratively estimate their trajectories in a common reference frame leveraging *distributed* computation. Collaborative PGO admits the same graph representations as collaborative rotation averaging (see Figure 2-1a). The main difference is that each vertex represents the full pose (instead of the rotation) of a robot at a certain time step. Odometry measurements and *intra-robot* loop closures connect poses within a single robot's trajectory. When two robots visit the same place (not necessarily at the same time), they establish *inter-robot* loop closures that link their respective poses [39–41]. The notions of separators and interior variables (Definition 2.1), as well as the robot-level dependency graph (Figure 2-1b) also apply for PGO.

2.2.3 Bundle Adjustment (BA)

Bundle adjustment (BA) is a fundamental problem in computer vision [42], and is a crucial building block of modern visual SLAM and SfM systems. While the problem formulation admits multiple variations, in this thesis, we focus on one of the most popular forms of BA that aims to jointly estimate 3D camera poses $\{T_1, \ldots, T_n\} \in \mathrm{SE}(3)^n$ and point landmarks $\{y_1, y_2, \ldots, y_m\} \in \mathbb{R}^{3\times m}$ from noisy 2D observations. Given known camera intrinsics, each observation $q_{jl} \in \mathbb{R}^2$ on the image plane relates

the pose of a camera $T_j = (R_j, t_j) \in \mathrm{SE}(3)$ and a 3D landmark $y_l \in \mathbb{R}^3$ as follows,

$$q_{jl} = \pi(T_j, y_l) + q_{jl}^{\mathrm{err}}, \;\; q_{jl}^{\mathrm{err}} \sim \mathcal{N}(0, w_{jl}^{-1} I_2). \tag{2.21}$$

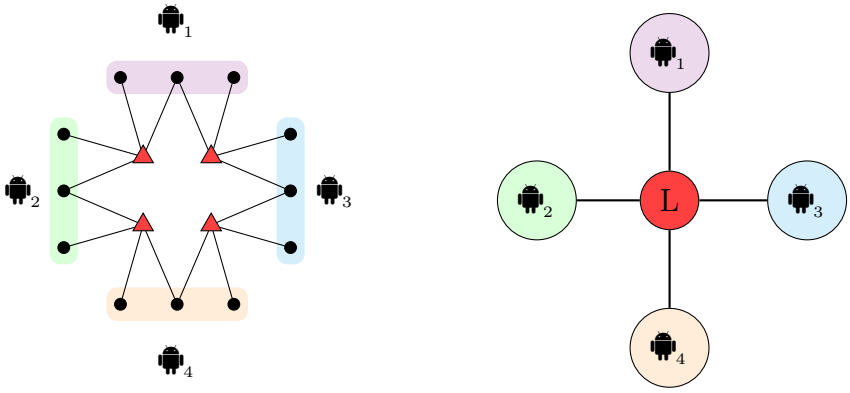
In (2.21), $\pi(T_j, y_l)$ is the camera reprojection function that predicts the 2D observation of landmark $y_l \in \mathbb{R}^3$ by a camera with pose $T_j \in \mathrm{SE}(3)$. The measurement noise $q_{jl} \in \mathbb{R}^2$ is assumed to be independently sampled from a Gaussian distribution on the image plane with covariance $\Sigma_{jl} = w_{jl}^{-1} I_2$. Under this measurement model, the maximum likelihood estimation formulation of BA is specified by the following optimization problem.

Problem 2.3 (Bundle Adjustment)**.**

$$\underset{\substack{T=(T_1,\ldots,R_n)\in\mathrm{SE}(3)^n,\\ y=(y_1,\ldots,y_m)\in\mathbb{R}^{3\times m}}}{\text{minimize}} \quad \sum_{(j,l)\in\mathcal{E}} w_{jl} \left\| q_{jl} - \pi(T_j, y_l) \right\|_2^2 . \tag{2.22}$$

Problem 2.3 also admit representation as a measurement graph $G = (\mathcal{V}, \mathcal{E})$. However, different from rotation averaging and PGO, the vertex set now contains two distinct types of vertices $\mathcal{V} = \mathcal{V}_T \uplus \mathcal{V}_p$ that correspond to the pose variables and landmark variables, respectively. Furthermore, G is a *bipartite graph*, because every edge $(j, l) \in \mathcal{E}$ connects a pose variable from $\mathcal{V}_T$ and a landmark variable from $\mathcal{V}_p$.

Collaborative BA. Chapter 7 studies BA in the setting of multi-robot collaborative estimation. Figure 2-2a illustrates an example BA measurement graph. Similar to the case of PGO, the overall set of pose variables $\mathcal{V}_T$ is partitioned into disjoint subsets, where each subset corresponds to the trajectory of a single robot. Furthermore, notice that the point landmarks $\mathcal{V}_p$ (shown as red triangle vertices) form a natural set of separators, given which robots' pose variables (black circle vertices) become disconnected. Figure 2-2b shows the corresponding robot-level dependency graph, where we treat the set of all landmarks as a single aggregated vertex (marked with "L"). Each robot vertex corresponds to the set of all pose variables owned by that robot. Note that robots' poses are only coupled together through the set of landmark



(a) Example measurement graph (b) Robot-level dependency graph

Figure 2-2: Graph representations for bundle adjustment. (a) Illustration of a simple BA measurement graph. Each robot has multiple pose variables (black circle vertices). The shared geometric map consists of multiple point landmarks (red triangle vertices) to be estimated. Each measurement (edge) relates a single pose variable and a single point landmark, and the overall graph is thus bipartite. (b) Robot-level dependency graph corresponding to the measurement graph in (a). Each robot vertex corresponds to the set of all pose variables owned by that robot. The middle vertex marked with "L" corresponds to the shared map (*i.e.*, the set of all point landmarks). In this case, the point landmarks form the separators and the dependency graph has a "star" structure.

variables. In Chapter 7, we exploit the special "star" graph structure in Figure 2-2b by developing a collaborative optimization algorithm in which a server updates the shared landmark map $\mathcal{V}_p$ and robots perform *parallel* and *distributed* updates over their sets of pose variables in $\mathcal{V}_T$.

Remark 2.1 (Extension to problem formulations beyond standard BA)**.** By employing alternative map representations and/or measurement models, the problem formulation in Problem 2.3 can be extended to many other estimation problems that belong to the general category of *feature-based* SLAM. All these formulations can be described with a factor graph [19]. Examples include other multi-view reconstruction problems that use alternative sensors (*e.g.*, lidar instead of camera) or estimate other types of geometric primitives (*e.g.*, planes, quadrics, and cuboids). However, in this thesis (in particular, in Chapter 7), we focus on the standard BA formulation (Problem 2.3) due to its fundamental role in multi-robot visual SLAM [1, 2, 4, 43].

Chapter 3

Literature Review

3.1 Collaborative SLAM

In this section, we review the crucial components that constitute a collaborative SLAM (CSLAM) system, including the front-end (inter-robot loop closure detection) and the back-end (localization and mapping). Then, we review state-of-the-art complete CSLAM systems and relevant datasets. The reader is also referred to [44] for a recent survey.

3.1.1 CSLAM Front-End

The CSLAM front-end is responsible for detecting inter-robot loop closures, which are critical to align the trajectories of the robots in a common reference frame and to improve their trajectory estimates. In a centralized visual SLAM system (*e.g.*, [2]), robots transmit a combination of global descriptors (*e.g.*, bag-of-words vectors [45, 46] and learned full-image descriptors [47]) and local visual features (*e.g.*, [48, 49]) to a central server that performs centralized place recognition and geometric verification. Recent work develops *distributed* and *communication-efficient* paradigms for inter-robot loop closure detection. Cieslewski and Scaramuzza [50] propose an efficient method for distributed visual place recognition, based on splitting and distributing bag-of-words visual features [45]. A subsequent approach is developed in [39, 51]

based on clustering and distributing NetVLAD [47] descriptors. A complementary line of work develops efficient methods for distributed geometric verification. Giamou *et al.* [40] develop a method to verify a set of candidate inter-robot loop closures using minimum data exchange. Tian *et al.* [41, 52] consider distributed geometric verification under communication and computation budgets and develop near-optimal communication policies based on submodular optimization.

3.1.2 CSLAM Back-End

The CSLAM back-end is responsible for performing the collaborative localization and mapping tasks given loop closures and data associations produced by the CSLAM front-end. This is usually formulated as a collaborative PGO or BA problem. Centralized CSLAM back-ends leverage off-the-shelf high-performance solvers such as GTSAM [6], Ceres Solver [7], and g2o [8]. Zhang *et al.* [53] develop a centralized incremental solver that generalizes the iSAM solver [54] to the setting of multi-agent estimation.

One of the earliest *distributed* SLAM solvers is DDF-SAM [55–57], in which each agent communicates a "condensed graph" produced by marginalizing out internal variables (those without inter-robot measurements) in its local Gaussian factor graph. Choudhary *et al.* [38] develop DGS, a two-stage approach for finding approximate solutions to multi-robot PGO in the distributed setting. The first stage approximately solves the underlying rotation averaging problem by relaxing the non-convex $\mathrm{SO}(d)$ constraints, solving the resulting (unconstrained) linear least squares problem, and projecting the results back to $\mathrm{SO}(d)$. The rotation estimates are then used in the second stage to initialize a single Gauss-Newton iteration on the full PGO problem. In both stages, iterative and distributable linear solvers such as Jacobi over-relaxation (JOR) and successive over-relaxation (SOR) [58] are used to solve the normal equations. The experimental evaluations presented in [38] demonstrate that this approach significantly outperforms prior techniques [55, 57]. Motivated by applications in camera network localization, Tron et al. [29, 29, 59] propose a multi-stage distributed estimation protocol based on distributed Riemannian gradient descent.

Camera network localization can be seen as a special instance of collaborative PGO where each agent owns a *single* pose rather than an entire trajectory. In these works, the authors establish convergence to critical points and, under *perfect* (noiseless) measurements, convergence to globally optimal solutions. Similar methods based on distributed Riemannian gradient descent include [60, 61]. Fan and Murphey [62–64] propose a majorization-minimization approach to solve distributed PGO. Each iteration constructs a quadratic upper bound on the cost function, and minimization of this upper bound is carried out in a distributed and parallel fashion. The core benefits of this approach are that it is *guaranteed* to converge to a first-order critical point of the PGO problem, and that it allows one to incorporate Nesterov's acceleration technique, which provides significant empirical speedup on typical PGO problems. Recently, Murai *et al.* [65] develop a new approach for distributed SLAM based on Gaussian Belief Propagation, and demonstrate its potential to scale to many agents during collaborative localization.

Related works in computer vision also propose to leverage distributed optimization to solve large-scale BA problems. Earlier work proposes to use distributed conjugate gradients for multi-core BA (*e.g.*, [66]). Methods of this type frequently leverage a domain decomposition approach (*e.g.*, see [67, Chapter 14]) that eliminates point landmarks from the normal equations using the Schur complement. Related works use sparse approximations of the resulting matrix (*e.g.*, with tree-based sparsity patterns) to precondition the optimization [66, 68–70]. More recently, researchers have proposed alternative algorithms for distributed BA based on Douglas-Rachford splitting [71] or alternating direction method of multipliers (ADMM) [72].

For collaborative mapping, existing works have explored different map representations, including dense geometric representations (*e.g.*, occupancy maps [73]) or sparse landmark maps [55, 57]; see [74] and the references therein. Recent work begins to incorporate sparse objects or dense semantic information in multi-robot perception. Choudhary *et al.* [38] use class labels to associate objects within a multi-robot pose graph SLAM framework. Tchuiev and Indelman [75] develop a distributed object-based SLAM method that leverages the coupling between object classification and

pose estimation. Yue *et al.* [76] leverage dense semantic segmentation to perform relative localization and map matching between pairs of robots.

3.1.3 Complete Systems

Existing CSLAM systems can be categorized based on whether they implement a *centralized* or *distributed* architecture. CCM-SLAM [2] is a well-established centralized system for visual-inertial CSLAM, in which a central server is responsible for multi-robot map management, fusion, and optimization. COVINS [4, 77] further extends and generalizes [2] and is demonstrated to scale to 12 robots. CVIDS [78] is another recent centralized CSLAM system that produces a dense global TSDF map. LAMP [3, 43] is a state-of-the-art centralized system for lidar-centric CSLAM and includes a loop closure prioritization module and an outlier-robust PGO module based on graduated non-convexity (GNC) [28]. MAPLAB 2.0 [5] is another recent centralized multi-robot multi-session SLAM system that supports multiple sensing modalities. While centralized systems offer great accuracy and ease of data management, they often require a stable connection with the server and are susceptible to a single point of failure.

Distributed CSLAM systems seek to alleviate the aforementioned limitations by removing the dependence on the central server. Zhang *et al.* [79] develop a distributed system for monocular-only CSLAM, where each robot performs global map merging onboard. Cieslewski *et al.* [39] and DOOR-SLAM [80] apply *distributed* PGO using the distributed Gauss-Seidel (DGS) method [38]. DOOR-SLAM [80] further employs Pairwise Consistency Maximization (PCM) [81] to reject outlier inter-robot loop closures. D^2SLAM [82] is a recent system that applies distributed and asynchronous optimization on multi-robot VIO and PGO. In parallel, Huang *et al.* [83] and Zhong *et al.* [84] also develop distributed systems for lidar-based CSLAM. Swarm-SLAM [85] is a very recent open-source system that supports both visual and lidar sensors. Building on a spectral sparsification method for single-robot SLAM [86], Swarm-SLAM prioritizes inter-robot loop closures by selecting candidates that maximize the algebraic connectivity of the multi-robot measurement graph. In contrast

to the distributed back-ends used by previous works [24, 39, 80], Swarm-SLAM implements a protocol that dynamically elects a leader among the connected robots to solve the full multi-robot PGO problem.

3.1.4 CSLAM Datasets

Apart from recent system works, several research groups have also contributed new large-scale CSLAM datasets. The NeBula [3] and CERBERUS [87] datasets are collected during the recent DARPA Subterranean Challenge. GRACO [88] includes multiple ground and aerial sequences for evaluating CSLAM using heterogeneous platforms. S3E [89] is a collection of CSLAM datasets that includes multiple indoor and outdoor trajectory designs with varying difficulties for 3 robots. M2DGR [90] are datasets collected by a single robot that traverses diverse scenarios, and contains multiple sequences in the same environments.

3.2 Certifiably Correct Geometric Estimation

Rosen *et al.* [14] developed SE-Sync, a state-of-the-art certifiably correct algorithm for PGO. SE-Sync is based upon a (convex) semidefinite relaxation that its authors prove admits a *unique, low-rank* minimizer providing an *exact, globally-optimal* solution to the original PGO problem whenever the noise on the available measurements is not too large; moreover, in the (typical) case that exactness obtains, it is possible to *verify* this fact *a posteriori* [91], thereby *certifying* the correctness (optimality) of the recovered estimate. To solve the resulting semidefinite program (SDP) efficiently, SE-Sync employs the Riemannian Staircase [92], which leverages symmetric low-rank (Burer-Monteiro) factorization [93] to directly search for a symmetric low-rank factor of the SDP solution, and implements this low-dimensional search using the truncated-Newton *Riemannian trust-region* (RTR) method [17, 94]. This combination of *low-rank factorization* and *fast local search* (via truncated-Newton RTR) enables SE-Sync to recover *certifiably globally optimal* PGO solutions at speeds comparable to (and frequently significantly faster than) standard state-of-the-art *local* search methods

(e.g. Gauss-Newton) [14].

A similar centralized solver, Cartan-Sync, is proposed in [20]. The main difference between SE-Sync and Cartan-Sync is that the latter directly relaxes the PGO problem *without* first analytically eliminating the translations [95]; consequently, the resulting relaxation retains the sparsity present in the original PGO problem. However, this alternative SDP relaxation (and consequently Cartan-Sync itself) has *not* previously been shown to enjoy any exactness guarantees; in particular, its minimizers, and their relation to solutions of PGO, have not previously been characterized. As one of the main contributions of Chapter 4, we derive sharp correspondences between minimizers of Cartan-Sync's relaxation and the original relaxation employed by SE-Sync (Theorem 4.1); in particular, this correspondence enables us to *extend* the exactness guarantees of the latter to cover the former, thereby justifying its use as a basis for the distributed certifiably correct PGO algorithms in Chapter 4.

As a related note, similar SDP relaxations [15, 96–98] have also been proposed for *rotation averaging* (Problem 2.3) [30]. Mathematically, rotation averaging can be derived as a *specialization* of PGO obtained by setting the measurement precisions for the translational observations to zero (practically, *ignoring* translational observations and states).

3.3 Outlier-Robust Geometric Estimation

Standard least squares formulation of PGO is susceptible to outlier loop closures that can severely impact trajectory estimation. To mitigate the effect of outliers in single-robot SLAM, early methods are based on RANSAC [99], branch & bound [100], and M-estimation ([30, 101]). Sünderhauf and Protzel [102] develop a method to deactivate outliers using binary variables. Agarwal *et al.* [103] build on the same idea and develop the dynamic covariance scaling method. Hartley *et al.* [104] and Casafranca *et al.* [105] propose to minimize the $\ell 1$-norm of residual errors. Chatterjee and Govindu [106, 107] develop iteratively reweighted least squares (IRLS) methods to solve rotation averaging using a family of robust cost functions. Hu *et al.* [108]

develop similar IRLS methods for single-robot SLAM. Olson and Agarwal [109] and Pfingsthorn and Birk [110, 111] consider multi-modal distributions for the noise. Lajoie *et al.* [112] and Carlone and Calafiore [113] develop global solvers based on convex relaxations. A separate line of work investigates consensus maximization formulations that seek to identify the maximal set of mutually consistent inliers [114–116]. Yang *et al.* [28] develop graduated non-convexity (GNC) that optimizes a sequence of increasingly non-convex surrogate cost functions, and demonstrate state-of-the-art performance on robust PGO problems. In addition to GNC as a fast heuristic for robust estimation, Yang and Carlone also develop methods with optimality certificates based on semidefinite relaxations and global optimization [16].

In multi-robot SLAM, Indelman *et al.* [117] and Dong *et al.* [118] apply expectation-maximization to find consistent inter-robot loop closures and estimate initial relative transformations between robots. Mangelson *et al.* [81] design the Pairwise Consistency Maximization (PCM) approach to perform robust map merging between pairs of robots. Lajoie *et al.* [80] implements an extended version of PCM as the outlier rejection method before distributed trajectory estimation.

3.4 Spectral Graph Theoretic Methods

3.4.1 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 [119–122] adopts an estimation-theoretic approach and shows that the Fisher information matrix is closely related to the underlying graph Laplacian matrix. Eriksson *et al.* [13] establish sufficient conditions for strong duality to hold in rotation averaging, where the derived analytical error bound depends on the algebraic connectivity of the graph. Doherty *et al.* [123] establish performance guarantees for spectral initialization in rotation averaging and PGO. Recently, Bernreiter *et al.* [124] use tools from graph signal processing to correct onboard estimation errors in multi-robot PGO.

From the perspective of nonlinear optimization, Carlone [125] analyzes the influences of graph connectivity and noise level on the convergence of Gauss-Newton methods when solving PGO. Tron [29] derives the Riemannian Hessian of rotation averaging under the geodesic distance, and uses the results to prove convergence of Riemannian gradient descent. In a pair of papers [126, 127], Wilson *et al.* study 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.* [128] develop 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.

3.4.2 Measurement Selection and Sparsification

Khosoussi *et al.* [120] and subsequent works [121, 122] propose measurement selection methods for geometric estimation based on the theory of optimal experimental design, where the objective is to maximize quantities such as the D-optimality criterion. Researchers have also developed information-based sparsification methods to sparsify the dense information matrix after marginalization, using methods such as the Chow-Liu tree (*e.g.*, [129, 130]) or convex optimization (*e.g.*, [131, 132]). 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. Recently, Doherty *et al.* [86] propose a measurement selection approach for pose graph SLAM that seeks to maximize the algebraic connectivity of the underlying graph.

3.4.3 Spectral Sparsification and Laplacian Solvers

Beyond the domain of geometric estimation considered in this thesis, the graph Laplacians also have immense applications in other fields such as control [133] and fast linear solvers [134]. Many of these applications leverage a common property of graph Laplacians, namely, they admit sparse approximations; see [135] for a survey. Spielman and Srivastava [136] 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.* [137] develop a procedure based on the so-called barrier functions for constructing *linear-sized* sparsifiers. Another line of work [138, 139] 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 [134] for a survey. Peng and Spielman [140] develop a parallel solver that invokes sparsification as a subroutine, which is improved and extended in following works [141, 142]. Recently, Tutunov [143] extends the approach in [140] to solve decentralized consensus optimization problems.

3.5 Block-Coordinate Descent Methods

Block-coordinate descent (BCD) methods (also known as Gauss-Seidel-type methods) are classical techniques [58, 144] that have recently regained popularity in large-scale machine learning and numerical optimization [145–148]. These methods are popular due to their simplicity, cheap iterations, and flexibility in the parallel and distributed settings [58]. BCD is a natural choice for solving PGO in the distributed setting due to the graphical decomposition of the underlying optimization problem. In fact, BCD-type techniques such as the Gauss-Seidel method have been applied in the past [149, 150] to solve SLAM. Similarly, in computer vision, variants of the Weiszfeld algorithm have also been used for robust rotation averaging [30, 104]. More recently, [98] propose a BCD-type algorithm for solving the SDP relaxation of rotation averaging. Their row-by-row (RBR) solver extends the approach of [151] from SDPs with diagonal constraints to block-diagonal constraints. The algorithms developed in Chapter 4 are originally inspired by block-coordinate minimization algorithms for solving SDPs with diagonal constraints via the Burer-Monteiro approach [152, 153]. Furthermore, the recent paper [154] extends these algorithms and the global convergence rate analysis

provided by [153] from the unit sphere (SDPs with diagonal constraints) to the Stiefel manifold (SDPs with *block*-diagonal constraints).

3.6 Asynchronous and Communication-Efficient Distributed Optimization

Within the broader optimization literature, there is a plethora of works on parallel and asynchronous optimization, partially motivated by popular applications in large-scale machine learning and deep learning. Study of asynchronous gradient-based algorithms began with the seminal work of Bertsekas and Tsitsilis [58], and has led to the recent development of asynchronous randomized block coordinate and stochastic gradient algorithms, see [155–161] and references therein. We are especially interested in asynchronous parallel schemes for non-convex optimization, which have been studied in [159, 161]. In Chapter 5, we generalize these approaches to the setting where the feasible set is the product of non-convex matrix manifolds, which makes our method applicable to distributed PGO applications.

Meanwhile, communication efficiency is another central theme in distributed optimization. Recently, this topic has gained increasing attention due to the success of federated learning [9–12]. Multiple techniques to achieve communication efficiency have been proposed, including the use of quantization [162] and distributed second-order methods [163]. In Chapter 7, we explore an alternative strategy based on *lazy* or *event-triggered* communication, which has demonstrated impressive results [164].

Chapter 4

Certifiably Correct Distributed Pose Graph Optimization

4.1 Introduction

Collaborative multi-robot missions require *consistent collective* spatial perception across the entire team. In unknown GPS-denied environments, this is achieved by *collaborative* simultaneous localization and mapping (CSLAM), in which a team of agents *jointly* constructs a *common* model of an environment via exploration. As introduced in Chapter 2, at the heart of CSLAM, robots must solve a *pose graph optimization* (PGO) problem to estimate their trajectories based on noisy relative *inter-robot* and *intra-robot* measurements.

While several prior approaches to CSLAM have appeared in the literature, to date no method has been proposed that is capable of *guaranteeing* the recovery of an optimal solution in the distributed setting. In this chapter, we advance the state of the art in CSLAM by proposing the first PGO algorithm that is both *fully distributed* and *certifiably correct*. Our method leverages the same semidefinite relaxation strategy that underpins current state-of-the-art (centralized) certifiably correct PGO algorithms [14], but employs novel decentralized optimization and solution verification techniques that enable these relaxations to be solved efficiently in the distributed setting.

Contributions. Specifically, this chapter makes the following contributions:

- We prove that a sparse semidefinite relaxation of PGO employed by [20] enjoys the same exactness guarantees as the one used in SE-Sync [14]: namely, that its minimizers are *low-rank* and provide *exact* solutions of the original PGO problem under moderate measurement noise.

- We describe an efficient low-rank optimization scheme to solve this semidefinite relaxation in the distributed setting. Specifically, we employ a *distributed* Riemannian Staircase approach [92], and propose *Riemannian block coordinate descent* (RBCD), a novel method for minimizing a function over a product of Riemannian manifolds, to solve the resulting low-rank subproblems in the distributed setting. We prove that RBCD converges to first-order critical points with a *global* sublinear rate under standard (mild) conditions, and that these are in particular *always* satisfied for the low-rank PGO subproblems. We also describe Nesterov-accelerated variants of RBCD that significantly improve its convergence speed in practice.

- We propose the first *distributed solution verification* and *saddle escape* methods to certify the optimality of low-rank critical points recovered via RBCD, and to descend from suboptimal critical points (if necessary).

- Finally, we describe simple distributed procedures for initializing the distributed Riemannian Staircase optimization, and for *rounding* the resulting low-rank factor to extract a final PGO estimate.

Each of these algorithmic components has the same communication, computation, and privacy properties enjoyed by current distributed CSLAM methods [38, 55, 57], including

1. *Communication and computational efficiency:* Robots need only communicate with their neighbors in the pose graph. To this end, the minimum requirement is that robots form a *connected* network, so that information can flow between any pair of robots (possibly relayed by intermediate robots). The payload size

in each round of communication is only $\mathcal{O}(m_{\text{inter}})$ where m_{inter} is the number of inter-robot loop closures. Moreover, local updates in RBCD can be performed efficiently and in *parallel*, and the solution is produced in an anytime fashion.

2. *Spatial privacy protection:* Robots are not required to reveal *any* information about their own observations or their *private* poses (those poses that are not *directly* observed by other robots).

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

The rest of this chapter is organized as follows. In the remainder of this section we introduce additional notations used in this chapter. In Section 4.2, we introduce the sparse SDP relaxation of PGO, and present a distributed procedure to solve this SDP via the Riemannian Staircase framework. On the theoretical front, we establish formal *exactness* guarantees for the SDP relaxation. In Section 4.3, we present our distributed local search method to solve the rank-restricted SDPs using block-coordinate descent. In Section 4.4, we prove convergence of the proposed local search method and analyze its global convergence rate. In Section 4.5, we present a distributed solution verification procedure that checks the global optimality of our local solutions, and enables us to *escape* from suboptimal critical points, if necessary. We discuss distributed initialization and rounding in Section 4.6. Finally, we conclude with extensive experimental evaluations in Section 4.7.

Notations

In the following, we introduce additional notations that are necessary for developing the algorithms in this chapter.

Linear Operators. Given a list of square matrices $A_1, \dots, A_n$ (possibly with different dimensions), Diag assembles them into a block-diagonal matrix:

$$\mathrm{Diag}(A_1, \dots, A_n) \triangleq \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_n \end{bmatrix}. \tag{4.1}$$

For an arbitrary square matrix A, Sym returns its symmetric part $\mathrm{Sym}(A) \triangleq (A + A^\top)/2$. Let $\mathrm{SBD}(d, n)$ denote the set of $[dn \times dn]$ symmetric block-diagonal matrices with diagonal blocks of size $d \times d$. For a $[d \times d]$-block-structured matrix $Z \in \mathbb{R}^{dn \times dn}$, we define the following linear operator that outputs another $[d \times d]$-block-diagonal matrix as follows,

$$\mathrm{SymBlockDiag}_d(Z)_{[i,j]} \triangleq \begin{cases} \mathrm{Sym}(Z_{[i,i]}), & \text{if } i = j, \\ 0_{d \times d}, & \text{otherwise.} \end{cases} \tag{4.2}$$

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

$$\mathrm{SymBlockDiag}_d^+(Z)_{[i,j]} \triangleq \begin{cases} \begin{bmatrix} \mathrm{Sym}(Z_{[i,i](1:d,1:d)}) & 0_d \\ 0_d^\top & 0 \end{bmatrix}, & \text{if } i = j, \\ 0_{(d+1)\times(d+1)}, & \text{otherwise.} \end{cases} \tag{4.3}$$

The PGO Manifold. In this chapter, the following product manifold is also used extensively and we give it a specific name:

$$\mathcal{M}_{\mathrm{PGO}}(r, n) \triangleq (\mathrm{St}(d, r) \times \mathbb{R}^r)^n \subset \mathbb{R}^{r \times (d+1)n}. \tag{4.4}$$

In our notation (4.4), we highlight the constants r and n and omit the constant d, as the latter is essentially fixed (*i.e.*, $d \in \{2, 3\}$ corresponding to 2D or 3D SLAM). As a product manifold, we can readily characterize the first-order geometry of $\mathcal{M}_{\mathrm{PGO}}(r, n)$

using those of $\mathrm{St}(d,r)$ and $\mathbb{R}^r$ (see Section 2.1 for an introduction to the Stiefel manifold). In particular, the tangent space of $\mathcal{M}_{\mathrm{PGO}}(r,n)$ is given as the Cartesian product of the tangent spaces of individual components. In matrix form, we can write the tangent space as:

$$T_X\mathcal{M}_{\mathrm{PGO}}(r,n) = \{\dot{X} \in \mathbb{R}^{r\times(d+1)n} \mid \mathrm{SymBlockDiag}_d^+(\dot{X}^\top X) = 0\}. \tag{4.5}$$

The normal space is the orthogonal complement of $T_X\mathcal{M}_{\mathrm{PGO}}(r,n)$ in the ambient space. It can be shown that the normal space takes the form,

$$T_X^\perp\mathcal{M}_{\mathrm{PGO}}(r,n) = \{XS \mid S = \mathrm{Diag}(S_1, 0, \ldots, S_n, 0), S_i \in \mathcal{S}^d\}. \tag{4.6}$$

Finally, given a matrix $U \in \mathbb{R}^{r\times(d+1)n}$ in the ambient space, the orthogonal projection onto the tangent space at X (4.5) is given by:

$$\begin{aligned}\mathrm{Proj}_{T_X} :&\mathbb{R}^{r\times(d+1)n} \to T_X\mathcal{M}_{\mathrm{PGO}}(r,n),\\ &U \mapsto U - X\,\mathrm{SymBlockDiag}_d^+(X^\top U).\end{aligned} \tag{4.7}$$

4.2 Certifiably Correct Pose Graph Optimization

Recall PGO (Problem 2.2) and its collaborative version presented in Section 2.2.2. In this section, we formally introduce the semidefinite relaxation of PGO and our certifiably correct algorithm for solving these in the distributed setting. Figure 4-1 summarizes the problems introduced in this section and how they relate to each other.

4.2.1 SDP Relaxation for PGO

Traditionally, Problem 2.2 is solved with local search algorithms such as Gauss-Newton. However, depending on the noise level and the quality of initialization, local search algorithms are susceptible to local minima [91]. To address this critical issue, recent works aim to develop *certifiably correct* PGO solvers. In particular, techniques based on SDP relaxation demonstrate empirical state-of-the-art performance

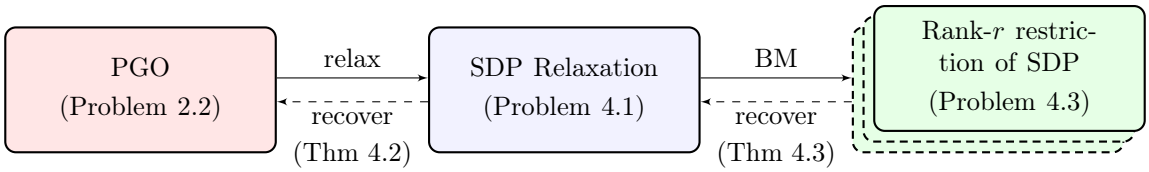


Figure 4-1: Relations between problems considered in Chapter 4. From the MLE formulation of PGO (Problem 2.2), applying semidefinite relaxation yields the SDP (Problem 4.1). Applying a distributed implementation of the Riemannian staircase algorithm [92] and Burer-Monteiro (BM) factorization [93] on the SDP then yields a set of rank-restricted problems (Problem 4.3) which can be *locally* optimized using our distributed Riemannian local search method (Section 4.3). After local search, the *global* optimality of the recovered first-order critical points of Problem 4.3 as solutions of the original SDP (Problem 4.1) can then be checked via *post hoc* verification, and if necessary, a descent direction can be constructed from a suboptimal critical point to continue the search (Section 4.5). Finally, under sufficiently low noise, SDP relaxations are guaranteed to recover *global* minimizers of PGO (Theorem 4.2).

while providing theoretical correctness (global optimality) guarantees under low noise regimes [14, 97, 98].

In this section, we present a semidefinite relaxation of Problem 2.2 that was first studied in [20]. Let $T \triangleq [R_1 \; t_1 \; \ldots \; R_n \; t_n] \in (\mathrm{SO}(d) \times \mathbb{R}^d)^n$ be the block-row matrix obtained by aggregating all rotation and translation variables. In [20], it is shown that the cost function in Problem 2.2 can be written in matrix form as $f(T) = \langle Q, T^\top T \rangle$, where $Q \in \mathcal{S}^{(d+1)n}$ is a symmetric matrix known as the *connection Laplacian* formed using all relative measurements. Consider the "lifted" variable $Z = T^\top T \in \mathcal{S}_+^{(d+1)n}$. Treating Z as a $(d+1)\times(d+1)$-block-structured matrix, we see that several necessary conditions for Z to satisfy the original constraints in PGO are,

$$Z \succeq 0, \tag{4.8}$$

$$\mathrm{rank}(Z) = d, \tag{4.9}$$

$$Z_{[i,i](1:d,1:d)} = R_i^\top R_i = I_{d\times d}, \; \forall i \in [n], \tag{4.10}$$

$$\det(Z_{[i,j](1:d,1:d)}) = \det(R_i^\top R_j) = 1, \; \forall i, j \in [n], i \neq j. \tag{4.11}$$

Dropping the non-convex rank and determinant constraints (4.9) and (4.11) yields an SDP relaxation of Problem 2.2.

Problem 4.1 (SDP Relaxation for Pose Graph Optimization [20])**.**

$$\underset{Z \in \mathcal{S}_+^{n+dn}}{\text{minimize}} \quad \langle Q, Z \rangle \tag{4.12a}$$

$$\text{subject to} \quad Z_{[i,i](1:d,1:d)} = I_{d\times d}, \forall i \in [n]. \tag{4.12b}$$

The original SE-Sync algorithm [14] employs a different SDP relaxation for Problem 2.2, obtained by first exploiting the so-called *separable* structure of PGO [95] to analytically eliminate the translation variables, and *then* performing convex relaxation over the resulting rotation-only problem. This approach yields:

Problem 4.2 (Rotation-only SDP Relaxation for Pose Graph Optimization [14])**.**

$$\underset{Z_{\mathrm{R}} \in \mathcal{S}_+^{dn}}{\text{minimize}} \quad \langle Q_{\mathrm{R}}, Z_{\mathrm{R}} \rangle \tag{4.13a}$$

$$\text{subject to} \quad Z_{\mathrm{R}[i,i]} = I_{d\times d}, \forall i \in [n], \tag{4.13b}$$

where Q_{R} is obtained by computing a generalized Schur complement of the connection Laplacian Q (Appendix A.1).

Remark 4.1 (Choosing the right SDP for Distributed PGO)**.** Problem 4.2 has several advantages over Problem 4.1, including a compact search space and better numerical conditioning. Nevertheless, unlike Q in Problem 4.1, the cost matrix Q_{R} in Problem 4.2 is generally *dense* (Appendix A.1.1). In graphical terms, eliminating the translation variables makes the underlying dependency graph *fully connected*. This is a major drawback in the distributed setting, since it corresponds to making all of the poses *public* (Definition 2.1), thereby substantially increasing the required communication. As we shall see in the following sections, our proposed algorithm relies on and exploits the *sparse* graphical structure (both intra-robot and inter-robot) of the problem to achieve computational and communication efficiency, and to preserve the privacy of participating robots. Therefore, in this work we seek to solve Problem 4.1 as a sparse convex relaxation to PGO.

However, in contrast to the SE-Sync relaxation (Problem 4.2) [14], Problem 4.1 has *not* previously been shown to enjoy any exactness guarantees. We now present new results to characterize the connection between the solutions of these problems, thereby *extending* the guarantee of exactness from Problem 4.2 to Problem 4.1.

Theorem 4.1 (Connection between Problems 4.1 and 4.2)**.** *Problem 4.1 admits a minimizer $Z^\star$ with* $\operatorname{rank}(Z^\star) = r$ if and only if *Problem 4.2 admits a minimizer $Z^\star_R$ with the same rank. Furthermore, $\langle Q, Z^\star \rangle = \langle Q_R, Z^\star_R \rangle$ for* all *minimizers of Problem 4.1 and Problem 4.2.*

Theorem 4.1 indicates that relaxing the additional translational variables when forming Problem 4.1 does not weaken the relaxation versus the SE-Sync relaxation (Problem 4.2), *nor* (crucially) introduce any additional minimizers that do *not* correspond to PGO solutions. In particular, Theorem 4.1 and [14, Proposition 2] together imply the following *exactness* guarantee for Problem 4.1 under low measurement noise.

Theorem 4.2 (Exact recovery via Problem 4.1)**.** *Let $\underline{Q}$ be the connection Laplacian in Problem 4.1 , constructed using the true (latent) relative transformations $(\underline{R}_{ij}, \underline{t}_{ij})$. There exists a constant $\delta > 0$ such that if $\left\|Q - \underline{Q}\right\|_2 < \delta$, every minimizer $Z^\star$ to Problem 4.1 has its first $d \times (n + dn)$ block row given by,*

$$Z^\star{}_{(1:d,:)} = \begin{bmatrix} R_1^\star & t_1^\star & \dots & R_n^\star & t_n^\star \end{bmatrix}, \tag{4.14}$$

where $\{R_i^\star, t_i^\star\}$ is an optimal solution to Problem 2.2.

Theorem 4.2 provides a crucial missing piece for achieving certifiably correct *distributed* PGO solvers: under low noise (quantified by the deviation in spectral norm of the connection Laplacian Q from its latent value), one can directly read off a global minimizer to PGO (Problem 2.2) from the first block row of any solution $Z^\star$ of the sparse SDP relaxation (Problem 4.1). As empirically shown in [14, 20], both SDP relaxations are exact in real-world scenarios (see Section 4.7 for additional empirical evidence).

4.2.2 Solving the Relaxation: The Distributed Riemannian Staircase

In typical CSLAM scenarios, the dimension of the SDP relaxation can be quite large (e.g. $\dim(Z) > 10^4$), and thus it is often impractical to solve Problem 4.1 using standard (interior-point) methods. In a seminal paper, [93] proposed a more scalable approach to search for *low-rank* solutions $Z^\star$ in particular: *assume* that some solution admits a symmetric low-rank factorization of the form $Z^\star = {X^\star}^\top X^\star$, where $X^\star \in \mathbb{R}^{r\times n}$ and $r \ll n$, and then directly search for the low-rank factor $X^\star$. This substitution has the two-fold effect of (i) dramatically reducing the dimension of the search space, and (ii) rendering the positive semidefiniteness constraint on Z *redundant*, since $X^\top X \succeq 0$ for *any* $X \in \mathbb{R}^{r\times n}$. In consequence, the *rank-restricted* version of the original semidefinite program obtained by performing the substitution $Z = X^\top X$ is actually a lower-dimensional *nonlinear program*, and so can be processed much more efficiently using standard (local) NLP methods.

For SDPs with block-diagonal constraints, [92] extends the general approach of [93] by further exploiting the *geometric* structure of the constraints in the Burer-Monteiro-factored problem. The result is an elegant algorithm known as the *Riemannian Staircase*, which is used to solve the (large-scale) semidefinite relaxations in SE-Sync [14] and Cartan-Sync [20].

In this work, we show how to implement the Riemannian Staircase approach in a *distributed* manner, thereby enabling us to solve collaborative PGO. Algorithm 4.1 presents our distributed Riemannian Staircase algorithm. In each iteration of the Riemannian Staircase, we assume a symmetric rank-r factorization $Z = X^\top X$ where $X \in \mathbb{R}^{r\times(n+dn)}$. Writing the blocks of X as $X = [Y_1\ p_1\ \dots\ Y_n\ p_n]$, the SDP constraints (4.12b) require that $Y_i \in \mathrm{St}(d,r)$ and $p_i \in \mathbb{R}^r$, for all $i \in [n]$. Equivalently, the aggregate variable X is constrained to live on the product manifold $\mathcal{M}_{\mathrm{PGO}}(r,n) \triangleq (\mathrm{St}(d,r)\times\mathbb{R}^r)^n$. Imposing the rank-$r$ factorization thus transforms the original SDP into the following *rank-restricted* problem:

Algorithm 4.1 DISTRIBUTED RIEMANNIAN STAIRCASE

Input:

- Initial point $X \in \mathcal{M}_{\text{PGO}}(r_0, n)$.

Output:

- A minimizer $X^\star \in \mathcal{M}_{\text{PGO}}(r, n)$ of Problem 4.3 such that $Z^\star = (X^\star)^\top (X^\star)$ is a solution to Problem 4.1.

1: **for** $r = r_0, r_0 + 1, \ldots$ **do**
2: Compute first-order critical point of Problem 4.3: $X^\star \leftarrow \text{RBCD}(X)$ (Section 4.3).
3: Lift to first-order critical point at next level: $X^\star \leftarrow [(X^\star)^\top \; 0]^\top$ as in (4.44).
4: Construct corresponding dual certificate matrix $S(X^\star)$ in (4.43).
5: Compute minimum eigenpair: $(\lambda, v) \leftarrow \text{MINEIG}(S(X^\star))$ (Algorithm 4.6).
6: **if** $\lambda \geq 0$ **then**
7: **Return** $X^\star$.
8: **else**
9: Construct second-order descent direction $\dot{X}_+$ in (4.45).
10: Descend from $X^\star$: $X \leftarrow \text{ESCAPESADDLE}(X^\star, \dot{X}_+)$ (Algorithm 4.7).
11: **end if**
12: **end for**

Problem 4.3 (Rank-restricted SDP for Pose Graph Optimization)**.**

$$\underset{X \in \mathcal{M}_{\text{PGO}}(r,n)}{\text{minimize}} \quad \langle Q, X^\top X \rangle. \tag{4.15}$$

In Section 4.3, we develop a novel local search algorithm, *Riemannian block coordinate descent* (RBCD), which we will use to recover first-order critical points of the rank-restricted SDP (Problem 4.3) in the distributed setting. Inspired by Nesterov's accelerated coordinate descent [148], we also propose an accelerated variant, RBCD++. In Section 4.4, we establish global first-order convergence guarantees for both RBCD and RBCD++.

From a first-order critical point $X^\star$, we ultimately wish to recover a solution to the SDP relaxation. To do so, we first lift $X^\star$ to the next level of the staircase (*i.e.*, increment rank r by one). This can be trivially done by padding $X^\star$ with a row of zeros (line 3). The motivations behind this operation will become clear later. By construction, the matrix $Z = X^{\star\top} X^\star$ is feasible in Problem 4.1. We may verify the global optimality of Z by checking the (necessary and sufficient) Karush-Kuhn-

Tucker (KKT) conditions; for a first-order critical point $X^\star$, this amounts to verifying that a certain dual certificate matrix $S(X^\star)$ is positive semidefinite (see line 4). In Section 4.5, we present the first distributed procedure to carry out this verification. If the dual certificate has a negative eigenvalue, then Z is *not* a minimizer of the SDP, and $X^\star$ is in fact a *saddle point* to Problem 4.3. Fortunately, in this case, the procedure in Section 4.5 also returns a descent direction, with which we can escape the saddle point (line 10) and restart distributed local search.

Remark 4.2 (First- vs. second-order optimization in the Riemannian Staircase)**.** The formulation of the Riemannian Staircase presented in Algorithm 4.1 differs *slightly* from its original presentation in [92]: specifically, the latter presupposes access to an algorithm that is capable of computing *second*-order critical points of Problem 4.3, whereas the Riemannian block coordinate descent method we employ in line 2 only guarantees convergence to *first*-order critical points. This has implications for the convergence properties of the overall algorithm: while one can show that the second-order version of the Riemannian Staircase [92, Alg. 1] is guaranteed to terminate at a level $r \leq n$ when applied to Problem 4.1 [92, Thm. 3.8],[1] the weaker (first-order) guarantees provided by RBCD are reflected in a correspondingly weaker set of convergence guarantees for our (first-order) Algorithm 4.1 provided in the following theorem.

Theorem 4.3 (Convergence of Algorithm 4.1)**.** *Let $\{X^{(r)}\}$ denote the sequence of low-rank factors generated by Algorithm 4.1 in line 3 using a particular saddle escape procedure described in Appendix A.3. Then exactly one of the following two cases holds:*

(*i*) *Algorithm 4.1 terminates after finitely many iterations and returns a symmetric factor $X^{(r)}$ for a minimizer $Z^\star = (X^{(r)})^\top X^{(r)}$ of Problem 4.1 in line 7.*

(*ii*) *Algorithm 4.1 generates an infinite sequence $\{X^{(r)}\}$ satisfying $f(X^{(r_2)}) < f(X^{(r_1)})$*

[1] Strictly speaking, the finite-termination guarantees provided by [92, Thm. 3.8] only hold for the SE-Sync relaxation Problem 4.2 (cf. [14, Prop. 3]); however, we can extend these guarantees to Problem 4.1 by exploiting the correspondence between critical points of the low-rank factorizations of Problems 4.1 and 4.2 that we establish in Lemma A.1 (Appendix A.1).

for all $r_2 > r_1$*, with*

$$\lim_{r\to\infty} f(X^{(r)}) = f^\star_{\text{SDP}}, \tag{4.16}$$

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

$$\lim_{k\to\infty} \lambda_{\min}\left(S(X^{(r_k)})\right) = 0. \tag{4.17}$$

In a nutshell, Theorem 4.3 states that Algorithm 4.1—with a particular version of saddle escape procedure described in Appendix A.3—either terminates after a finite number of iterations, or generates an infinite sequence of factors $\{X^{(r)}\}$ that monotonically strictly decrease the objective to the optimal value $f^\star_{\text{SDP}}$ *and* that can arbitrarily well-approximate the satisfaction of the KKT condition $\lambda_{\min}(S(X^{(r)})) \geq 0$. We prove this theorem in Appendix A.3.

We remark that while the convergence guarantees of Theorem 4.3 are *formally* weaker than those achievable using a second-order local search method, as a *practical* matter these differences are inconsequential. In any numerical implementation of the Riemannian Staircase framework, both the second-order criticality of a stationary point (in the second-order version) and the nonnegativity of the minimum eigenvalue λ (in Algorithm 4.1) are checked subject to some numerical tolerance $\epsilon_{\text{tol}} > 0$; this accounts for both the finite precision of real-world computers, and the fact that the low-rank factors X computed via local search in line 2 are *themselves* only *approximations* to critical points, as they are obtained using iterative local optimization methods. In particular, practical implementations of Algorithm 4.1 (including ours) would replace line 6 with a termination condition of the form "$\lambda \geq -\epsilon_{\text{tol}}$"[2], and (4.17) guarantees that this condition is satisfied after *finitely* many iterations for *any* $\epsilon_{\text{tol}} > 0$. As a practical matter, the behavior of Algorithm 4.1 is far from the pessimistic case described in part (*ii*) of Theorem 4.3; as we show empirically in Section 4.7, in real-world applications typically only 1-3 iterations suffice.

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

Algorithm 4.2 Distributed Certifiably Correct Pose Graph Optimization (DC2-PGO)

Input:

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

Output:

- A feasible solution $T \in \mathrm{SE}(d)^n$ to Problem 2.2 and the lower bound $f^\star_{\mathrm{SDP}}$ on Problem 2.2's optimal value.

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

4.2.3 The Complete Algorithm

The distributed Riemannian Staircase (Algorithm 4.1) is the core computational procedure of our overall algorithm. Nevertheless, to implement a *complete* distributed method for solving the original PGO problem (Problem 2.2), we must still specify procedures for (i) initializing the Riemannian Staircase by constructing an initial point $X \in \mathcal{M}_{\mathrm{PGO}}(r_0, n)$, and (ii) *rounding* the low-rank factor $X^\star$ returned by the Riemannian Staircase to extract a feasible solution $T \in \mathrm{SE}(d)^n$ of the PGO problem. We discuss the details of both distributed initialization and rounding in Section 4.6. Combining these procedures produces our complete distributed certifiably correct algorithm, DC2-PGO (Algorithm 4.2).

Since the SDP (Problem 4.1) is a convex relaxation of PGO, its optimal value $f^\star_{\mathrm{SDP}}$ is necessarily a lower bound on the global minimum of PGO. Using this fact, we may obtain an *upper* bound on the suboptimality of the solution returned by DC2-PGO. Specifically, let $f(T)$ denote the objective achieved by the final estimate, and let $f^\star_{\mathrm{MLE}}$ denote the optimal value of Problem 2.2. Then:

$$f(T) - f^\star_{\mathrm{MLE}} \leq f(T) - f^\star_{\mathrm{SDP}}. \tag{4.18}$$

In particular, if $f(T) = f^\star_{\mathrm{SDP}}$, then the SDP relaxation is *exact*, and $f(T) = f^\star_{\mathrm{MLE}}$. In this case, (4.18) serves as a *certificate* of the *global* optimality of T.

4.3 Distributed Local Search via Riemannian Block-Coordinate Descent

In this section, we introduce a new *distributed* local search algorithm to identify a first-order critical point of the rank-restricted SDP relaxation (Problem 4.3), which is needed by the Distributed Riemannian Staircase framework (Algorithm 4.1, line 2). Our algorithm is applicable to a broad class of smooth optimization problems defined over the Cartesian product of matrix manifolds:

$$\underset{X\in\mathcal{M}}{\text{minimize}} \quad f(X), \quad \mathcal{M} \triangleq \mathcal{M}_1 \times \ldots \times \mathcal{M}_N. \tag{4.19}$$

The above problem contains Problem 4.3 as a special case. Specifically, in distributed PGO, each block b exactly corresponds to a robot and $\mathcal{M}_b = \mathcal{M}_{\text{PGO}}(r, n_b) \triangleq (\text{St}(d, r)\times \mathbb{R}^r)^{n_b}$ corresponds to the search space of this robot's trajectory. Here, n_b is the number of poses owned by the robot associated to block b, and N is the total number of robots. For this reason, unless otherwise mentioned, in this section we use the words "block" and "robot" interchangeably.

To solve (4.19), we leverage the product structure of the underlying manifold and propose a distributed *block-coordinate descent* algorithm that we call RBCD (Algorithm 4.3). In each iteration of RBCD, a block $b \in [N]$ is selected to be optimized. Specifically, let $X_b \in \mathcal{M}_b$ be the component of X corresponding to the selected block, and let $\widehat{X}_{[N]\setminus\{b\}}$ be the (fixed) values of remaining blocks. We update X_b by minimizing the following *reduced cost function*.

$$\underset{X_b\in\mathcal{M}_b}{\text{minimize}} \quad f_b(X_b) \triangleq f(X_b, \widehat{X}_{[N]\setminus\{b\}}). \tag{4.20}$$

For the rank-restricted SDP (Problem 4.3) in PGO, the reduced problem for block b takes the form,

$$f_b(X_b) = \langle Q_b, X_b^\top X_b\rangle + 2\langle F_b, X_b\rangle + \text{const.} \tag{4.21}$$

In the above equation, Q_b is the submatrix of Q formed with the rows and columns

that correspond to block b (i.e., the trajectory of robot b), and $F_b \in \mathbb{R}^{r\times(d+1)n_b}$ is a constant matrix that depends on the (fixed) public variables of robot b's neighbors in the pose graph.

Remark 4.3 (Communication requirements of RBCD)**.** RBCD is designed such that it can be easily implemented by a network of robots. At each iteration, the team first coordinates to select the next block (robot) to update (Algorithm 4.3, line 3). Then, to update the selected block (Algorithm 4.3, line 4), the robot corresponding to this block receives public variables from its neighboring robots in the pose graph. Afterwards, this robot forms and solves its local optimization problem (4.20), which does not require further communications. Finally, to determine when to terminate RBCD (Algorithm 4.3, line 2), robots need to collaboratively evaluate their total gradient norm. In practice, checking the termination condition may be done periodically (instead of after every iteration) to save communication resources.

Remark 4.4 (Block-coordinate minimization on product manifolds)**.** Prior works (e.g., [152–154]) have proposed similar *block-coordinate minimization* (BCM) algorithms to solve low-rank factorizations of SDPs with diagonal or block-diagonal constraints. Our approach generalizes these methods in two major ways. First, while prior methods are explicitly designed for problems over the product of spheres [152, 153] or Stiefel manifolds [154], our algorithm is applicable to the product of *any* matrix manifolds. Secondly, prior works [152–154] require that the cost function to have a certain quadratic form, so that exact minimization of each variable block admits a closed-form solution. In contrast, our algorithm does not seek to perform exact minimization, but instead computes an inexpensive *approximate* update that achieves a *sufficient reduction* of the objective (see Section 4.3.2). This makes our method more general and applicable to a much broader class of smooth cost functions that satisfy a Lipschitz-type condition. We discuss this point in greater detail in Section 4.4.

The rest of this section is organized to discuss each step of RBCD in detail. We begin in Section 4.3.1 by discussing *block selection rules*. These rules determine

Algorithm 4.3 Riemannian Block-Coordinate Descent (RBCD)

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

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

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

how blocks are selected at each iteration (Algorithm 4.3, line 3). In Section 4.3.2, we propose a general *block update rule* (Algorithm 4.3, line 4) based on approximate minimization of trust-region subproblems [94]. In Section 4.3.3, we further develop an *accelerated* variant of RBCD based on Nesterov's celebrated accelerated coordinate-descent algorithm [148], which greatly speeds up convergence near critical points. Finally, in Section 4.3.4 we show that, using a slight modification of our blocking scheme, we can allow multiple robots to update their coordinates in parallel, thereby speeding up our distributed local search.

4.3.1 Block Selection Rules

In this section, we describe three mechanisms for selecting which block to update at each iteration of RBCD (Algorithm 4.3, line 3). We note that similar rules have been proposed in the past; see, e.g., [145, 153, 165].

- **Uniform Sampling**. The first rule is based on the idea of uniform sampling. At each iteration, each block $b \in [N]$ is selected with equal probability $p_b = 1/N$.
- **Importance Sampling**. In practice, it is often the case that selecting certain blocks leads to significantly better performance compared to others [145]. Therefore, it is natural to assign these blocks higher weights during the sampling

process. We refer to this block selection rule as *importance sampling*. In this work, we set the probability of selecting each block to be proportional to the squared gradient norm, i.e., $p_b \propto \|\text{grad}_b f(X)\|^2, \forall b \in [N]$. Here, $\text{grad}_b f(X)$ denotes the component of the Riemannian gradient of $f(X)$ that corresponds to block b. Under Lipschitz-type conditions, the squared gradient norm can be used to construct a lower bound on the achieved cost decrement; see Lemma 4.1.

- **Greedy (Gauss-Southwell)**. We can also modify importance sampling into a deterministic strategy that simply selects the block with the largest squared gradient norm, i.e., $b \in \arg\max \|\text{grad}_b f(X)\|^2$. We refer to this strategy as *greedy selection* or the *Gauss-Southwell (GS)* rule [145]. Recent works also propose other variants of greedy selection such as Gauss-Southwell-Lipschitz (GSL) and Gauss-Southwell-Quadratic (GSQ) [145]. However, such rules require additional knowledge about the block Lipschitz constants that are hard to obtain in our application. For this reason, we restrict our deterministic selection rule to GS. Despite its simplicity, empirically the GS rule exhibits satisfactory performance; see Section 4.7.

Remark 4.5 (Communication requirements of different block selection rules)**.** In practice, uniform sampling does not incur communication overhead, and can be approximately implemented using synchronized clocks on each robot (to conduct and coordinate BCD rounds) and a common random seed for the pseudorandom number generator (to agree on which robot should update in the next round). In contrast, importance sampling and greedy selection require additional communication overhead at each round, as robots need to evaluate and exchange local gradient norms. In particular, the greedy selection rule can be implemented via flooding gradient norms; see, e.g., the FloodMax algorithm for leader election in general synchronized networks [166, Chapter 4]. This requires robots to have unique IDs and communicate in synchronized rounds. While greedy and importance rules have higher communication overhead than uniform sampling, they also produce more effective iterations and thus converge faster (see Section 4.7).

4.3.2 Computing a Block Update

Algorithm 4.4 BlockUpdate

Input:

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

Output:

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

1: $\Delta \leftarrow \Delta_0$.
2: Form model function $\hat{m}_b(\eta_b) = f_b(X_b^k) + \langle \operatorname{grad} f_b(X_b^k), \eta_b \rangle + \frac{1}{2}\langle \eta_b, H[\eta_b] \rangle$.
3: **while** true **do**
4: Compute an *approximate* solution $\eta_b^\star \in T_{X_b^k}\mathcal{M}_b$ to the trust-region subproblem (4.24).
5: **if** $\rho(\eta_b^\star) > 1/4$ **then**
6: **return** $X_b^{k+1} = \operatorname{Retr}_{X_b^k}(\eta_b^\star)$.
7: **else**
8: Decrease trust-region radius $\Delta \leftarrow \Delta/4$.
9: **end if**
10: **end while**

Note that since (4.20) is in general a non-convex minimization, computing a block update by *exactly* solving this problem is intractable. In this section, we describe how to implement a cheaper approach that permits the use of *approximate* solutions of (4.20) by requiring only that they produce a *sufficient decrease* of the objective. In Section 4.4, we show that under mild conditions, such approximate updates are sufficient to ensure global first-order convergence of RBCD. While there are many options to achieve sufficient descent, in this work we propose to (approximately) solve a single trust-region subproblem using the truncated preconditioned conjugate gradient method [17, 94]. Compared to a full minimization that would solve (4.20) to first-order critical point, our approach greatly reduces the computational cost. On the other hand, unlike other approximate update methods such as Riemannian gradient descent, our method allows us to leverage (local) second-order information of the reduced cost which leads to more effective updates.

Let b be the block that we select to update, and denote the current value of this

block (at iteration k) as X_b^k. We define the *pullback* of the reduced cost function (4.20) as follows [17, 167],

$$\begin{aligned} \widehat{f}_b : T_{X_b^k} &\to \mathbb{R}, \\ \eta_b &\mapsto f_b \circ \mathrm{Retr}_{X_b^k}(\eta_b). \end{aligned} \tag{4.22}$$

Note that the pullback is conveniently defined on the tangent space which itself is a vector space. However, since directly minimizing the pullback is nontrivial, it is approximated with a quadratic *model* function [17, 167], as defined below.

$$\widehat{m}_b(\eta_b) \triangleq f_b(X_b^k) + \langle \mathrm{grad}\, f_b(X_b^k), \eta_b \rangle + \frac{1}{2}\langle \eta_b, H[\eta_b] \rangle. \tag{4.23}$$

In (4.23), $H : T_{X_b^k} \to T_{X_b^k}$ is a user-specified mapping on the tangent space. By default, we use the Riemannian Hessian $H = \mathrm{Hess}\, f_b(X_b^k)$ so that the model function is a second-order approximation of the pullback. Then, we compute an update direction $\eta_b^\star$ on the tangent space by approximately solving the following trust-region subproblem,

$$\mathrm{minimize}_{\eta_b \in T_{X_b^k}\mathcal{M}_b} \ \widehat{m}_b(\eta_b) \ \text{ subject to } \ \|\eta_b\| \leq \Delta. \tag{4.24}$$

To ensure that the obtained update direction yields sufficient descent on the original pullback, we follow standard procedure [17, 94] and evaluate the following ratio that quantifies the *agreement* between model decrease (predicted reduction) and pullback decrease (actual reduction),

$$\rho(\eta_b^\star) \triangleq \frac{\widehat{f}_b(0) - \widehat{f}_b(\eta_b^\star)}{\widehat{m}_b(0) - \widehat{m}_b(\eta_b^\star)} = \frac{f_b(X_b^k) - \widehat{f}_b(\eta_b^\star)}{f_b(X_b^k) - \widehat{m}_b(\eta_b^\star)}. \tag{4.25}$$

If the above ratio is larger than a constant threshold (default to 1/4), we accept the current update direction and set $X_b^{k+1} = \mathrm{Retr}_{X_b^k}(\eta_b^\star)$ as the updated value of this block. Otherwise, we reduce the trust-region radius Δ and solve the trust-region subproblem again. Algorithm 4.4 gives the pseudocode for the entire block update procedure.

In Appendix A.2, we prove that under mild conditions, we can always find an

update direction (the so-called *Cauchy step* [17, 94]) that satisfies the required termination condition (Algorithm 4.4, line 5). Furthermore, the returned solution is guaranteed to produce *sufficient descent* on the cost function, which is crucial to establish global convergence rate of RBCD (Algorithm 4.3). We discuss the details of convergence analysis in Section 4.4.

Remark 4.6 (Solving the trust-region subproblem (4.24))**.** Following [17, 94], we also use the truncated conjugate-gradient (tCG) method to solve the trust-region subproblem (4.24) inside Algorithm 4.4. tCG is an efficient "inverse-free" method, i.e., it does not require inverting the Hessian itself, and instead only requires evaluating Hessian-vector products. Furthermore, tCG can be significantly accelerated using a suitable *preconditioner*. Formally, a preconditioner is a linear, symmetric, and positive-definite operator on the tangent space that approximates the inverse of the Riemannian Hessian. Both SE-Sync [14, 168] and Cartan-Sync [20] have already proposed empirically effective preconditioners for problems similar to (4.21).[3] Drawing similar intuitions from these works, we design our preconditioner as,

$$\begin{aligned} \operatorname{Precon} f(X_b^k) : T_{X_b^k} &\to T_{X_b^k}, \\ \eta_b &\mapsto \operatorname{Proj}_{T_{X_b^k}}(\eta_b (Q_b + \lambda I)^{-1}). \end{aligned} \tag{4.26}$$

The small constant $\lambda > 0$ ensures that the proposed preconditioner is positive-definite. In practice, we can store and reuse the Cholesky decomposition of $Q_b + \lambda I$ for improved numerical efficiency.

Remark 4.7 (Block update via exact minimization)**.** Algorithm 4.4 employs RTR [94] to *sufficiently reduce* the cost function along a block coordinate. It is worth noting that in some special cases, one can *exactly* minimize the cost function along any single block coordinate [153, 154]. In the context of PGO, this is the case when every block consists only of a *single* pose which naturally arises in camera network localization. We provide a quick sketch in the following. First, note that the reduced

[3] The only difference is the additional linear terms in our cost functions, as a result of anchoring variables owned by other robots.

problem (4.21) is an unconstrained convex quadratic over the Euclidean component of $\mathcal{M}_b = \mathrm{St}(d, r) \times \mathbb{R}^r$ (i.e., the so-called *lifted* translation component). We can thus first eliminate this component analytically by minimizing the reduced cost over the lifted translation vector, thereby further reducing (4.21) to an optimization problem over $\mathrm{St}(d, r)$. Interestingly, the resulting problem too admits a closed-form solution via the projection operator onto $\mathrm{St}(d, r)$ provided in (2.10) (see also [154, Sec. 2] for a similar approach). Finally, using this solution we can recover the optimal value for the Euclidean component via linear least squares (see, e.g., [95]).

4.3.3 Accelerated Riemannian Block-Coordinate Descent

In practice, many PGO problems are poorly conditioned. Critically, this means that a generic first-order algorithm can suffer from slow convergence as the iterates approach a first-order critical point. Such slow convergence is also manifested by the typical *sublinear* convergence rate, e.g., for Riemannian gradient descent as shown in [167]. To address this issue, Fan and Murphy [62, 63] recently developed a majorization-minimization algorithm for PGO. Crucially, their approach can be augmented with a generalized version of Nesterov's acceleration that significantly speeds up empirical convergence.

Following the same vein of ideas, we show that it is possible to significantly speed up RBCD by adapting the celebrated accelerated coordinate-descent method (ACDM), originally developed by Nesterov [148] to solve smooth convex optimization problems. Compared to the standard randomized coordinate descent method, ACDM enjoys an accelerated convergence rate of $\mathcal{O}(1/k^2)$. Let N denote the dimension (number of coordinates) in the problem. ACDM updates two scalar sequences

Algorithm 4.5 Accelerated Riemannian Block-Coordinate Descent (RBCD++)

Input:

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

Output:

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

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

$\{\gamma_k\}, \{\alpha_k\}$ and three sequences of iterates $\{x^k\}, \{y^k\}, \{v^k\} \in \mathbb{R}^N$.

$$\gamma_k = (1 + \sqrt{1 + 4N^2{\gamma_{k-1}}^2})/2N, \tag{4.27}$$

$$\alpha_k = 1/(\gamma_k N), \tag{4.28}$$

$$y^k = (1 - \alpha_k)x^k + \alpha_k v^k, \tag{4.29}$$

$$x^{k+1} = y^k - 1/L_{b_k} \nabla_{b_k} f(y^k), \tag{4.30}$$

$$v^{k+1} = v^k + \gamma_k(x^{k+1} - y^k). \tag{4.31}$$

In (4.30), L_{b_k} is the Lipschitz constant of the gradient that corresponds to coordinate b_k. Note that compared to standard references (e.g., [144, 148]), we have slightly changed the presentation of ACDM, so that later it can be extended to our Riemannian setting in a more straightforward manner. Still, it can be readily verified that (4.27)-(4.31) are equivalent to the original algorithm.[4]

In Algorithm 4.5, we adapt the ACDM iterations to design an accelerated variant of RBCD, which we call RBCD++. We leverage the fact that our manifolds of interest are naturally embedded within some linear space. This allows us to first perform the additions and subtractions as stated in (4.27)-(4.31) in the linear ambient space, and subsequently *project* the result back to the manifold. For our main manifold of interest $\mathcal{M}_{\text{PGO}}(r, n)$, the projection operation only requires computing the SVD for each Stiefel component, as shown in (2.10). Note that the original ACDM method performs a coordinate descent step (4.30) at each iteration. In RBCD++, we generalize (4.30) by employing the BlockUpdate procedure (Algorithm 4.4) to perform a descent step on a *block coordinate* $Y_b \in \mathcal{M}_b$ (Algorithm 4.5, line 8).

Unlike the convex case, it is unclear how to prove convergence of the above acceleration scheme subject to the non-convex manifold constraints. Fortunately, convergence can be guaranteed by adding *adaptive restart* [169], which has also been employed in recent works [62, 63]. The underlying idea is to ensure that each RBCD++ update (specifically on the $\{X^t\}$ variables) yields a *sufficient reduction* of the overall cost function. This is quantified by comparing the descent with the squared gradient norm at the selected block (Algorithm 4.5, line 13), where the constant $c_1 > 0$ specifies the minimum amount of descent enforced at each iteration. If this criterion is met, the algorithm simply continues to the next iteration. If not, the algorithm switches to the default block update method (same as RBCD), and restarts the acceleration scheme from scratch. Empirically, we observe that setting c_1 close to zero (corresponding to a permissive acceptance criterion) gives the best performance.

Remark 4.8 (Adaptive vs. fixed restart schemes)**.** Our adaptive restart scheme

[4]For example, we can recover (4.27)-(4.31) from [144, Algorithm 4], by setting the strong convexity parameter σ to zero.

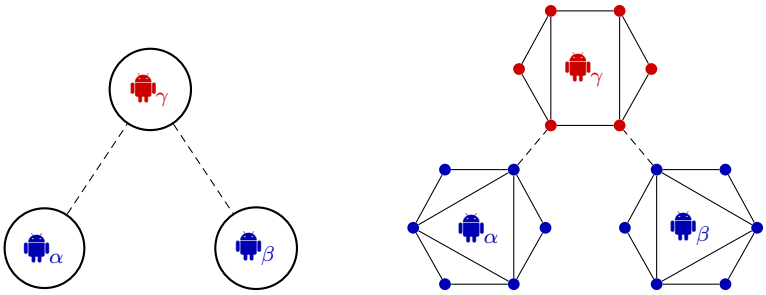
requires aggregating information from all robots to evaluate the cost function and gradient norm (Algorithm 4.5, line 13). This step may become the communication bottleneck of the whole algorithm. While in theory we need adaptive restart to guarantee convergence (see Section 4.4), a practical remedy is to employ a *fixed restart* scheme [169] whereby we simply restart acceleration (Algorithm 4.5, lines 14-19) periodically in fixed intervals. Our empirical results in Section 4.7 show that the fixed restart scheme also achieves significant acceleration, although is inferior to adaptive restart scheme.

Remark 4.9 (Communication requirements of RBCD++)**.** With fixed restart, the communication pattern of RBCD++ is identical to RBCD. In particular, with synchronized clocks, robots can update the scalars γ_k and α_k (line 3) locally in parallel. Similarly, the "Y update" (line 5) and "V update" steps (line 11) do not require communication, since both only involve local linear combinations and projections to manifold. The main communication happens before the "X update" (line 8), where each robot communicates the public components of their Y variables with neighbors in the global pose graph. Finally, if adaptive restart is used, robots need to communicate and aggregate global cost and gradient norms to evaluate the restart condition (line 13).

4.3.4 Parallel Riemannian Block-Coordinate Descent

Thus far in each round of RBCD and RBCD++ (Algorithms 4.3 and 4.5), exactly one robot performs BlockUpdate (Algorithm 4.4). However, after a slight modification of our blocking scheme, *multiple* robots may update their variables *in parallel* as long as they are *not* neighbours in the dependency graph (i.e., do not share an inter-robot loop closure; see Figure 2-1b). This is achieved by leveraging the natural graphical decomposition of objectives in Problem 4.3 (inherited from Problem 2.2). Updating variables in parallel can significantly speed up the local search.

Gauss-Seidel-type updates can be executed in parallel using a classical technique known as red-black coloring (or, more generally, multicoloring schemes) [58]. We



(a) A 2-coloring for the dependency graph

(b) Induced aggregate blocks

Figure 4-2: Parallel updates for the collective pose graph shown in Figure 2-1: (a) First, we find a coloring for the corresponding dependency graph such that adjacent robots have different colors; (b) The 2-coloring induces two "aggregate blocks" $\mathcal{A}_1, \mathcal{A}_2$ where $\mathcal{A}_1$ and $\mathcal{A}_2$ consist of all blue and red vertices, respectively. In each iteration, we select a color and update the corresponding variables. Note that $\mathcal{A}_1$ contains variables from both α and β, and therefore these robots can update their variables in parallel when blue is selected.

apply this technique to PGO (Figure 4-2):

1. First, we find a coloring for the set of *robots* such that *adjacent robots* in the dependency graph have different colors (Figure 4-2a). Although finding a vertex coloring with the *smallest* number of colors is NP-hard, simple greedy approximation algorithms can produce a $(\Delta + 1)$-coloring, where Δ is the maximum degree of the dependency graph; see [170, 171] and the references therein for distributed algorithms. Note that Δ is often bounded by a small constant due to the sparsity of the CSLAM dependency graph.

2. In each iteration, we select a color (instead of a single robot) by adapting the block selection rules presented in Section 4.3.1. The robots that have the selected color then update their variables in parallel.

Implementing the (generalized) importance sampling and greedy rules with coloring requires additional coordination between the robots. In particular, the greedy rule requires computing the sum of squared gradient norms for each color at the beginning of each iteration. Similar to Section 4.3.1, a naïve approach would be to flood the network with the current squared gradient norms such that after a sufficient number of rounds (specifically, the diameter of the dependency graph), every robot

aggregates all squared gradient norm information for every color. Robots can then independently compute the sum of squared gradient norms for every color and update their block only if their color has the largest gradient norm among all colors. We conclude this part by noting that it is also possible to allow *all* robots to update their *private* variables in *all* iterations (irrespective of the selected color) because private variables are separated from each other by public variables.

4.4 Convergence Analysis for RBCD and RBCD++

In this section, we formally establish first-order convergence guarantees for RBCD (Algorithm 4.3) and its accelerated variant RBCD++ (Algorithm 4.5), for generic optimization problems of the form (4.19) defined on the Cartesian product of smooth matrix submanifolds. Then, at the end of this section, we remark on how our general convergence guarantees apply when solving the rank-restricted SDPs of PGO (Remark 4.10), with detailed proofs and discussions in Appendix A.2.4. Our global convergence proofs extend the recent work of Boumal et al. [167]. Similar to [167], we begin by listing and discussing several mild technical assumptions.

Assumption 4.1 (Lipschitz-type gradient for pullbacks)**.** In the optimization problem (4.19), consider the reduced cost f_b and its pullback $\widehat{f}_b$ for an arbitrary block $b \in [N]$. There exists a constant $c_b \geq 0$ such that at any iterate X^k generated by a specified algorithm, the following inequality holds for all $\eta_b \in T_{X_b^k}\mathcal{M}_b$,

$$\left|\widehat{f}_b(\eta_b) - [f_b(X_b^k) + \langle \eta_b, \operatorname{grad} f_b(X_b^k)\rangle]\right| \leq \frac{c_b}{2}\left\|\eta_b\right\|_2^2. \tag{4.32}$$

In [167], Boumal *et al.* use (4.32) as a convenient generalization of the classical property of Lipschitz continuous gradient. With this property, the authors show that it is straightforward to establish global first-order convergence guarantees, e.g., for the well-known Riemannian gradient descent algorithm. Furthermore, it is shown that (4.32) holds under mild conditions in practice, e.g., whenever the cost function

has Lipschitz continuous gradients in the ambient Euclidean space and the underlying submanifold is compact [167, Lemma 2.7]. In Appendix A.2.4, we will generalize this result to show that Assumption 4.1 holds when running RBCD and RBCD++ in the context of PGO.

Assumption 4.2 (Global radial linearity of H)**.** In BlockUpdate (Algorithm 4.4), the user-specified map H (4.23) is globally radially linear, i.e., for any block $b \in [N]$,

$$H[c\eta_b] = cH[\eta_b], \text{ for all } \eta_b \in T_{X_b}\mathcal{M}_b \text{ and } c \geq 0. \tag{4.33}$$

Assumption 4.3 (Boundedness of H)**.** In BlockUpdate (Algorithm 4.4), the user-specified map H (4.23) is bounded along the iterates of local search, i.e., there exists $c_0 \geq 0$ such that for at any iterate X^k generated by a specified algorithm, the following inequality holds for any block $b \in [N]$,

$$\max_{\eta_b \in T_{X_b^k}\mathcal{M}_b, \|\eta_b\| \leq 1} |\langle \eta_b, H[\eta_b] \rangle| \leq c_0. \tag{4.34}$$

Assumption 4.4 (Lower bound on initial trust-region radius)**.** In BlockUpdate (Algorithm 4.4), the initial trust-region radius Δ_0 is bounded away from zero by,

$$\Delta_0 \geq \lambda_b \left\| \operatorname{grad}_{X_b} f_b \right\|, \tag{4.35}$$

where X_b is the input block estimate to Algorithm 4.4 and λ_b is a block-specific constant defined as,

$$\lambda_b \triangleq \frac{1}{8(c_b + c_0)}. \tag{4.36}$$

Assumptions 4.2-4.4 concern the execution of BlockUpdate (Algorithm 4.4), and are once again fairly lax in practice. In particular, the simplest choice of H that

satisfies radial linearity (4.33) and boundedness (4.34) is the identity mapping. In our PGO application, we use the Riemannian Hessian $H = \operatorname{Hess} f_b(X_b)$ to leverage local second-order information for faster convergence. In this case, H is still radially linear, and in Appendix A.2.4 we show that H is also bounded along the sequence of iterates generated by RBCD or RBCD++. Finally, Assumption 4.4 can be easily satisfied by using a sufficiently large initial trust-region radius. We are now ready to establish an important theoretical result, which states that each block update (Algorithm 4.4) yields *sufficient decrease* on the corresponding reduced cost function.

Lemma 4.1 (Sufficient descent property of Algorithm 4.4)**.** *Under Assumptions 4.1-4.4, applying* BlockUpdate*(Algorithm 4.4) on a block $b \in [N]$ with an input value X_b^k decreases the reduced cost by at least,*

$$f_b(X_b^k) - f_b(X_b^{k+1}) \geq \frac{1}{4}\lambda_b \left\|\operatorname{grad} f_b(X_b^k)\right\|^2, \tag{4.37}$$

where λ_b is the block-specific constant corresponding to the selected block defined in (4.36)*.*

Lemma 4.1 states that after each iteration of RBCD, we are guaranteed to decrease the corresponding reduced cost function. Furthermore, the amount of reduction is *lower bounded* by some constant times the squared gradient norm at the selected block. Thus, if we execute RBCD for long enough, intuitively we should expect the iterates to converge to a solution where the gradient norm is zero (i.e., a first-order critical point). The following theorem formalizes this result.

Theorem 4.4 (Global convergence rate of RBCD)**.** *Let $f^\star$ denote the global minimum of the optimization problem* (4.19)*. Denote the iterates of RBCD (Algorithm 4.3) as $X^0, X^1, \ldots, X^{K-1}$, and the corresponding block selected at each iteration as $b_0, \ldots, b_{K-1}$. Under Assumptions 4.1-4.4, RBCD with* uniform sampling *or* importance sampling *have the following guarantees,*

$$\min_{0\leq k\leq K-1} \mathbb{E}_{b_{0:k-1}} \left\|\operatorname{grad} f(X^k)\right\|^2 \leq \frac{4N(f(X^0) - f^\star)}{K \cdot \min_{b\in[N]} \lambda_b}. \tag{4.38}$$

In addition, RBCD with greedy selection *yields the following* deterministic *guarantee,*

$$\min_{0\leq k\leq K-1} \left\|\operatorname{grad} f(X^k)\right\|^2 \leq \frac{4N(f(X^0)-f^\star)}{K\cdot \min_{b\in[N]}\lambda_b}. \tag{4.39}$$

Theorem 4.4 establishes a *global* sublinear convergence rate for RBCD.[5] Specifically, as the number of iterations K increases, the squared gradient norm decreases at the rate of $\mathcal{O}(1/K)$. Using the same proof technique, we can establish a similar convergence guarantee for the accelerated version RBCD++.

Theorem 4.5 (Global convergence rate of RBCD++)**.** *Let $f^\star$ denote the global minimum of the optimization problem* (4.19)*. Denote the iterates of RBCD++ (Algorithm 4.5) as $X^0, X^1, \ldots, X^{K-1}$, and the corresponding block selected at each iteration as $b_0, \ldots, b_{K-1}$. Define the constant,*

$$C \triangleq \min\left(c_1, \min_{b\in[N]} \lambda_b/4\right). \tag{4.40}$$

Under Assumptions 4.1-4.4, RBCD++ with uniform sampling *or* importance sampling *have the following guarantees,*

$$\min_{0\leq k\leq K-1} \mathbb{E}_{b_{0:k-1}} \left\|\operatorname{grad} f(X^k)\right\|^2 \leq N(f(X^0)-f^\star)/CK. \tag{4.41}$$

In addition, RBCD++ with greedy selection *yields the following* deterministic *guarantee,*

$$\min_{0\leq k\leq K-1} \left\|\operatorname{grad} f(X^k)\right\|^2 \leq N(f(X^0)-f^\star)/CK. \tag{4.42}$$

Remark 4.10 (Convergence on Problem 4.3)**.** So far, we have shown that under mild conditions, RBCD and RBCD++ are guaranteed to converge when solving general op-

[5]We note that in our current analysis, uniform sampling and importance sampling share the same convergence rate estimate. In practice, however, it is usually the case that importance sampling yields much faster empirical convergence (see Section 4.7). This result suggests that it is possible to further improve the convergence guarantees for importance sampling. We leave this for future work.

timization problems defined on the Cartesian product of smooth matrix submanifolds. Recall that in the specific application of PGO, we are using RBCD and RBCD++ to solve a sequence of rank-restricted semidefinite relaxations (Problem 4.3). In Appendix A.2.4, we show that Assumptions 4.1-4.4 are satisfied in this case, and hence RBCD and RBCD++ retain their respective convergence guarantees. In particular, we show that although Assumption 4.1 (Lipschitz-type gradient) and Assumption 4.3 (bounded Hessian) do not hold globally (due to the non-compact translation search space), they still hold within the *sublevel set* of the cost function determined by the initial iterate X^0. Since RBCD and RBCD++ are inherently *descent* methods, it suffice to restrict our attention to this initial sublevel set as future iterates will not leave this set. The reader is referred to Appendix A.2.4 for more details.

Remark 4.11 (Convergence under parallel executions)**.** With the parallel iterations described in Section 4.3.4, we can further improve the global convergence rates of RBCD and RBCD++ in Theorems 4.4 and 4.5. For example, the constant N that appears in the rate estimates of uniform sampling and importance sampling (4.38) can be replaced by the number of aggregate blocks (colors) in the dependency graph. Recall that in practice, this is typically bounded by a small constant, e.g., the maximum degree in the sparse robot-level dependency graph (see Section 4.3.4).

4.5 Distributed Verification

In this section we address the problem of *solution verification* [91] in the distributed setting. Concretely, we propose distributed solution verification and saddle escape algorithms to certify the optimality of a first-order critical point X of the rank-restricted relaxation (4.15) as a global minimizer $Z = X^\top X$ of problem (4.12), and for *escaping* from suboptimal stationary points after ascending to the next level of the Riemannian Staircase (Algorithm 4.1). To the best of our knowledge, these are the first distributed solution verification algorithms to appear in the literature.

Our approach is based upon the following simple theorem of the alternative, which is a specialization of [172, Theorem 4] to problems (4.12) and (4.15):

Theorem 4.6 (Solution verification and saddle escape)**.** *Let $X \in \mathcal{M}_{\mathrm{PGO}}(r,n)$ be a first-order critical point of the rank-restricted semidefinite relaxation* (4.15)*, and define:*

$$\Lambda(X) \triangleq \mathrm{SymBlockDiag}_d^+(X^\top X Q), \tag{4.43a}$$

$$S(X) \triangleq Q - \Lambda(X). \tag{4.43b}$$

Then exactly one of the following two cases holds:

(*a*) *$S(X) \succeq 0$ and $Z = X^\top X$ is a global minimizer of* (4.12)*.*

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

$$X_+ \triangleq \begin{bmatrix} X \\ 0 \end{bmatrix} \in \mathcal{M}_{\mathrm{PGO}}(r+1,n) \tag{4.44}$$

is a first-order critical point of (4.15) *attaining the same objective value as X, and*

$$\dot{X}_+ \triangleq \begin{bmatrix} 0 \\ v^\top \end{bmatrix} \in T_{X_+}(\mathcal{M}_{\mathrm{PGO}}(r+1,n)) \tag{4.45}$$

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

Remark 4.12 (Interpretation of Theorem 4.6)**.** Let us provide a bit of intuition for what Theorem 4.6 conveys. Part (a) is simply the standard (necessary and sufficient) conditions for $Z = X^\top X$ to be the solution of the (convex) semidefinite program (4.12) [173]. In the event that these conditions are *not* satisfied (and therefore Z is *not* optimal in (4.12)), there must exist a direction of descent $\dot{Z} \in \mathcal{S}^{(d+1)n}$ from Z that is *not* captured in the low-rank factorization (4.15), *at least to first order* (since X is stationary). This could be because X is a saddle point of the non-convex problem (4.15) (in which case there may exist a *second*-order direction of descent from X), or because the descent direction $\dot{Z}$ is towards a set of higher-rank matrices than the rank-r factorization used in (4.15) is able to capture. Part (b) of Theorem 4.6 provides an

approach that enables us to address *both* of these potential obstacles simultaneously, by using a negative eigenvector of the certificate matrix $S(X)$ to construct a *second*-order direction of descent $\dot{X}_+$ from X_+, the lifting of X to the next (higher-rank) "step" of the Riemannian Staircase. Geometrically, this construction is based upon the (easily verified) fact that $S(X)$ is the Hessian of the Lagrangian $\nabla^2_X \mathcal{L}$ of the extrinsic (constrained) form of (4.15), and therefore $\langle \dot{X}_+, \nabla^2_X \mathcal{L}\, \dot{X}_+ \rangle = \langle v, S(X)v \rangle < 0$, so that $\dot{X}_+$ is indeed a direction of second-order descent from the lifted stationary point X_+ [172, 174, 175].

In summary, Theorem 4.6 enables us to determine whether a first-order critical point X of (4.15) corresponds to a minimizer $Z = X^\top X$ of (4.12), and to *descend* from X if necessary, by computing the minimum eigenpair (λ, v) of the certificate matrix $S(X)$ defined in (4.43). In the original SE-Sync algorithm, the corresponding minimum-eigenvalue computation is performed by means of spectrally-shifted Lanczos iterations [14, 168]; while this works well for the centralized SE-Sync method, adopting the Lanczos algorithm in the distributed setting would require an excessive degree of communication among the agents. Therefore, in the next subsection, we investigate alternative strategies for computing the minimum eigenpair that are more amenable to a distributed implementation.

4.5.1 Distributed Minimum-eigenvalue Computation

In this subsection we describe an efficient distributed algorithm for computing the minimum eigenpair of the certificate matrix $S(X)$ required in Theorem 4.6. We begin with a brief review of eigenvalue methods.

In general, the Lanczos procedure is the preferred technique for computing a small number of extremal (maximal or minimal) eigenpairs of a symmetric matrix $A \in \mathcal{S}^n$ [176, Chp. 9]. In brief, this method proceeds by approximating A using its orthogonal projection $P_k \triangleq V_k^\top A V_k$ onto the *Krylov subspace*:

$$\mathcal{K}(A, x_0, k) \triangleq \operatorname{span}\left\{x_0, Ax_0, \ldots, A^{k-1}x_0\right\} = \operatorname{image} V_k, \tag{4.46}$$

where $x_0 \in \mathbb{R}^n$ is an initial vector and $V_k \in \mathrm{St}(k,n)$ is a matrix whose columns (called *Lanczos vectors*) provide an orthonormal basis for $\mathcal{K}(A, x_0, k)$. Eigenvalues θ_i of the (low-dimensional) approximation P_k, called *Ritz values*, may then be taken as approximations for eigenvalues of A. The k-dimensional Krylov subspace $\mathcal{K}(A, x_0, k)$ in (4.46) is iteratively expanded as the algorithm runs (by computing additional matrix-vector products), thereby providing increasingly better approximations of A's extremal eigenvalues (in accordance with the Courant-Fischer variational characterization of eigenvalues [176, Thm. 8.1.2]). The Lanczos procedure thus provides an efficient means of estimating a subset of A's spectrum (particularly its extremal eigenvalues) to high accuracy at the cost of only a relatively small number (compared to A's dimension n) of matrix-vector products, especially if the initial vector x_0 lies close to an eigenvector of A. In particular, if $\lambda(A) = \lambda_1 > \lambda_2 > \cdots > \lambda_n$ and ϕ_1 is the eigenvector associated to λ_1, it is well-known that the error in the eigenvector estimate y_1 from the maximal Ritz pair (θ_1, y_1) decays asymptotically according to [177, eq. (2.15)]:

$$\sin(\phi_1, y_1) \sim \tau_1^{-k}, \tag{4.47}$$

where

$$\tau_1 = \rho_1 + \sqrt{\rho_1^2 - 1}, \tag{4.48a}$$

$$\rho_1 = 1 + 2\frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}. \tag{4.48b}$$

However, while the Lanczos procedure is the method of choice for computing a few extremal eigenpairs in the *centralized* setting, it is unfortunately not well-suited to *distributed* computations when inter-node communication is a bottleneck. This is because the Lanczos vectors V_k must be periodically re-orthonormalized in order to preserve the accuracy of the estimated eigenpairs (θ_i, v_i). While several strategies have been proposed for performing this reorthonormalization, all of them essentially involve computing a QR decomposition of (possibly a subset of columns from) V_k (see [176, Sec. 9.2] and the references therein). Constructing this decomposition in the distributed setting would require frequent synchronized all-to-all message passing,

which is impractical when inter-node communication is expensive or unreliable.

We are therefore interested in exploring alternatives to the Lanczos method that require less coordination in the distributed setting. Many commonly-used eigenvalue methods can be viewed as attempts to simplify the "gold-standard" Lanczos procedure, achieving a reduction in storage and per-iteration computation at the cost of a slower convergence rate. An extreme example of this is the well-known *power method* [176, Sec. 8.2], which can be viewed as a simplification that retains only the final generator $A^{k-1}x_0$ of the Krylov subspace $\mathcal{K}(A, x_0, k)$ in (4.46) at each iteration. This leads to a very simple iteration scheme, requiring only matrix-vector products:[6]

$$x_{k+1} = Ax_k. \tag{4.49}$$

Note that with $A = S(X)$, the matrix-vector product in (4.49) can be computed using the same inter-agent communication pattern already employed in each iteration of the RBCD method developed in Section 4.3, and so is well-suited to a distributed implementation. However, the power method's simplicity comes at the expense of a reduced convergence rate. In particular, if λ_1 and λ_2 are the two largest-magnitude eigenvalues of A (with $|\lambda_1| > |\lambda_2|$), then the vector y_1 in the Ritz estimate (θ_1, y_1) computed via the power method converges to the dominant eigenvector ϕ_1 of A according to [176, Thm. 8.2.1]:

$$\sin(\phi_1, y_1) \sim \left|\frac{\lambda_1}{\lambda_2}\right|^{-k}. \tag{4.50}$$

Let us compare the rates (4.47) and (4.50) for the case in which the eigengap $\gamma \triangleq \lambda_1 - \lambda_2$ is small relative to the diameter D of the spectrum of A (which is of the same order as $\lambda_2 - \lambda_n$ and λ_2 for the Lanczos and power methods, respectively); intuitively, this is the regime in which the problem is hard, since it is difficult to distinguish λ_1 and λ_2 (and consequently their associated eigenspaces). In particular,

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

let us compute the number of iterations $k(\epsilon)$ required to reduce the angular error in the dominant Ritz vector y_1 by the factor $\epsilon \in (0,1)$. For the Lanczos method, using (4.47) and (4.48) and assuming $\gamma \ll D$, we estimate:

$$k_l(\epsilon) = -\frac{\log \epsilon}{\log \tau_1} \approx -\frac{\log \epsilon}{\log\left(1 + 2\frac{\gamma}{D} + \sqrt{4\frac{\gamma}{D} + 4\frac{\gamma^2}{D^2}}\right)} \approx -\frac{\log \epsilon}{\log\left(1 + 2\sqrt{\frac{\gamma}{D}}\right)} \approx -\frac{\log \epsilon}{2\sqrt{\frac{\gamma}{D}}}. \tag{4.51}$$

Similarly, using (4.50), the analogous estimate for the power method is:

$$k_p(\epsilon) = -\frac{\log \epsilon}{\log\left(\frac{\lambda_1}{\lambda_2}\right)} = -\frac{\log \epsilon}{\log\left(1 + \frac{\gamma}{\lambda_2}\right)} \approx -\frac{\log \epsilon}{\gamma/D}. \tag{4.52}$$

In our target application (certifying the optimality of a first order-critical point X), the minimum eigenvalue we must compute will *always* belong to a tight cluster whenever X is a global minimizer,[7] so the power method's $\mathcal{O}(1/\gamma)$ dependence upon the eigengap γ translates to a substantial reduction in performance versus the Lanczos method's $\mathcal{O}(1/\sqrt{\gamma})$ rate.

In light of these considerations, we propose to adopt the recently-developed *accelerated power method* [178] as our distributed eigenvalue algorithm of choice. In brief, this method modifies the standard power iteration (4.49) by adding a Polyak momentum term, producing the iteration:

$$x_{k+1} = Ax_k - \beta x_{k-1}, \tag{4.53}$$

where $\beta \in \mathbb{R}_+$ is a *fixed* constant. We note that because β is constant, the iteration (4.53) has the same communication pattern as the standard power method (4.49), and so is well-suited to implementation in the distributed setting. Furthermore, despite the simplicity of the modification (4.53) versus (4.49), the addition of momentum actually allows the accelerated power method to *match* the $\mathcal{O}(1/\sqrt{\gamma})$ dependence of the Lanczos method on the dominant eigengap for a well-chosen parameter β. More

[7] This is an immediate consequence of the (extrinsic) first-order criticality condition for (4.15), which requires $S(X)X^\top = 0$, i.e., that *each row* of X be an eigenvector of $S(X)$ with eigenvalue 0 [168, Sec. III-C].

precisely, we have the following result:

Theorem 4.7 (Theorem 8 of [178]). *Let $A \in \mathcal{S}_+^n$ with eigenvalues $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, and ϕ_1 be the eigenvector associated with the maximum eigenvalue λ_1. Given $\beta \in \mathbb{R}_+$ satisfying $\beta < \lambda_1^2/4$ and an initial Ritz vector $y_0 \in \mathbb{R}^n$, after k accelerated power iterations* (4.53) *the angular error in the Ritz vector y_k satisfies:*

$$\sin(\phi_1, y_k) \leq \frac{\sqrt{1-(\phi_1^\top y_0)^2}}{|\phi_1^\top y_0|} \cdot \begin{cases} 2\left(\frac{2\sqrt{\beta}}{\lambda_1+\sqrt{\lambda_1^2-4\beta}}\right)^k & \beta > \lambda_2^2/4 \\ \left(\frac{\lambda_2+\sqrt{\lambda_1^2-4\beta}}{\lambda_1+\sqrt{\lambda_1^2-4\beta}}\right)^k & \beta \leq \lambda_2^2/4 \end{cases}. \tag{4.54}$$

Remark 4.13 (Selection of β)**.** While the hypotheses of Theorem 4.7 require that $\beta < \lambda_1^2/4$, note that lower bounds on λ_1 (which provide admissible values of β) are easy to obtain; indeed, the Courant-Fischer theorem [176, Thm. 8.1.2] implies that $\lambda_1 \geq y^\top A y$ for *any* unit vector $y \in \mathbb{R}^n$. We also observe that the bound on the right-hand side of (4.54) is an *increasing* function of β for $\beta > \lambda_2^2/4$ and a *decreasing* function for $\beta \leq \lambda_2^2/4$, with

$$\begin{aligned} \lim_{\beta \to (\lambda_2^2/4)^+} \frac{2\sqrt{\beta}}{\lambda_1+\sqrt{\lambda_1^2-4\beta}} &= \frac{\lambda_2}{\lambda_1+\sqrt{\lambda_1^2-\lambda_2^2}} \\ < \frac{\lambda_2+\sqrt{\lambda_1^2-\lambda_2^2}}{\lambda_1+\sqrt{\lambda_1^2-\lambda_2^2}} &= \lim_{\beta \to (\lambda_2^2/4)^-} \frac{\lambda_2+\sqrt{\lambda_1^2-4\beta}}{\lambda_1+\sqrt{\lambda_1^2-4\beta}}; \end{aligned} \tag{4.55}$$

consequently, the optimal rate is achieved in the limit $\beta \to (\lambda_2^2/4)^+$.

Combining the power and accelerated power methods with the spectral shifting strategy proposed in [168, Sec. III-C] produces our distributed minimum-eigenvalue method (Algorithm 4.6). In brief, the main idea is to construct a spectrally-shifted version C of the certificate matrix $S(X)$ such that (i) a maximum eigenvector v of C coincides with a *minimum* eigenvector of S, and (ii) $C \succeq 0$, so that we can recover the maximum eigenpair (θ, v) of C using accelerated power iterations (4.53). Algorithm 4.6 accomplishes this by first applying the (basic) power method (4.49) to estimate the dominant eigenpair $(\lambda_{\text{dom}}, v_{\text{dom}})$ of S in line 1 (which does *not* require $S \succeq 0$),

Algorithm 4.6 MINIMUM EIGENPAIR (MINEIG)

Input: Certificate matrix $S = S(X)$ from (4.43b).
Output: Minimum eigenpair $(\lambda_{\min}, v_{\min})$ of S.

1: Compute dominant (maximum-magnitude) eigenpair $(\lambda_{\text{dom}}, v_{\text{dom}})$ of S using power iteration (4.49).
2: **if** $\lambda_{\text{dom}} < 0$ **then**
3: **return** $(\lambda_{\text{dom}}, v_{\text{dom}})$
4: **end if**
5: Compute maximum eigenpair (θ, v) of $C \triangleq \lambda_{\text{dom}} I - S$ using accelerated power iteration (4.53).
6: **return** $(\lambda_{\text{dom}} - \theta, v)$

and then applying the accelerated power method to compute the maximum eigenpair (θ, v) of $C = \lambda_{\text{dom}} I - S \succeq 0$ in line 5. Note that while the minimum eigenvalue of $S(X)$ belongs to a tight cluster whenever X is optimal for (4.15) (necessitating our use of *accelerated* power iterations in line 5), the *dominant* eigenvalue of S is typically well-separated, and therefore can be computed to high precision using only a small number of power iterations in line 1.

Remark 4.14 (Communication requirements of Algorithm 4.6)**.** The bulk of the work in Algorithm 4.6 lies in updating the eigenvector estimate via the matrix-vector products (4.49) and (4.53). In the distributed regime, these can be implemented by having each robot estimate the block of the eigenvector that corresponds to its own poses. The communication pattern of this process is determined by the sparsity structure of the underlying matrix, which for our application are the dual certificate S and its spectrally-shifted version C. Fortunately, both S and C inherit the sparsity of the connection Laplacian Q. This means that at each iteration of (4.49) or (4.53), each robot only needs to communicate with its neighbors in the global pose graph. Therefore, Algorithm 4.6 provides an efficient way (in terms of *both* computation *and* communication) to compute a minimum eigenpair of S in the distributed setting.

4.5.2 Descent from Suboptimal Critical Points

In this subsection we describe a simple procedure for *descending* from a first-order critical point $X \in \mathcal{M}_{\text{PGO}}(r, n)$ of Problem 4.3 and restarting local optimization in the

event that $Z = X^\top X$ is *not* a minimizer of Problem 4.1 (as determined by $\lambda < 0$, where (λ, v) is the minimum eigenpair of the certificate matrix $S(X)$ in Theorem 4.6).

In this setting, Theorem 4.6(b) shows how to use the minimum eigenvector v of $S(X)$ to construct a *second*-order direction of descent $\dot{X}_+$ from the lifting X_+ of X to the next level of the Riemannian Staircase. Therefore, we can descend from X_+ by performing a simple backtracking line-search along $\dot{X}_+$; we summarize this procedure as Algorithm 4.7. Note that since $\operatorname{grad} f(X_+) = 0$ and $\langle \dot{X}_+, \operatorname{Hess} f(X_+)[\dot{X}_+] \rangle < 0$ by Theorem 4.6(b), letting $X(\alpha) \triangleq \operatorname{Retr}_{X_+}(\alpha \dot{X}_+)$, there exists a stepsize $\delta > 0$ such that $f(X(\alpha)) < f(X_+)$ and $\|\operatorname{grad} f(X(\alpha))\| > 0$ for all $0 < \alpha < \delta$, and therefore the loop in line 3 is guaranteed to terminate after finitely many iterations. Algorithm 4.7 is thus well-defined. Moreover, since α decreases at an exponential rate (line 4), in practice only a handful of iterations are typically required to identify an acceptable stepsize. Therefore, even though Algorithm 4.7 requires coordination among all of the agents (to evaluate the objective $f(X(\alpha))$ and gradient norm $\|\operatorname{grad} f(X(\alpha))\|$ each trial point $X(\alpha)$, and to distribute the trial stepsize α), it requires a sufficiently small number of (very lightweight) globally-synchronized messages to remain tractable in the distributed setting. Finally, since the point returned by Algorithm 4.7 has nonzero gradient, it provides a nonstationary initialization for local search at the next level $r + 1$ of the Riemannian Staircase (Algorithm 4.1), thereby enabling us to continue the search for a low-rank factor in Problem 4.3 corresponding to a *global* minimizer of the SDP relaxation Problem 4.1.

4.6 Distributed Initialization and Rounding

4.6.1 Distributed Initialization

A distinguishing feature of our approach versus prior distributed PGO methods is that it enables the direct computation of *globally* optimal solutions of PGO Problem 2.2 via (convex) semidefinite programming, and therefore does *not* depend upon a high-quality initialization in order to recover a good solution. Nevertheless, it can still

Algorithm 4.7 DESCENT FROM A SUBOPTIMAL CRITICAL POINT X_+ (ESCAPESADDLE)

Input:
- Lifted suboptimal critical point X_+ as defined in (4.44).
- Second-order descent direction $\dot{X}_+$ as defined in (4.45).

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

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

benefit (in terms of reduced computation time) from being supplied with a high-quality initial estimate whenever one is available.

Arguably the simplest method of constructing such an initial estimate is *spanning tree initialization* [179]. As the name suggests, we compute the initial pose estimates by propagating the noisy pairwise measurements along an arbitrary spanning tree of the global pose graph. In the distributed scenario, this technique incurs minimal computation and communication costs, as robots only need to exchange few public poses with their neighbors.

While efficient, the spanning tree initialization is heavily influenced by the noise of selected edges in the pose graph. A more resilient but also more heavyweight method is *chordal initialization*, originally developed to initialize rotation synchronization [31, 180]. With this technique, one first relaxes rotation synchronization into a linear least squares problem, and subsequently projects the solution back to the rotation group. For distributed computation, [38] propose to solve the resulting linear least squares problem via distributed iterative techniques such as the Jacobi and Gauss-Seidel methods [58]. One detail to note is that in [38], the translation estimates are not explicitly initialized but are instead directly optimized during a single distributed Gauss-Newton iteration. However, we find that this approach leads to poor convergence for [38] on some real-world datasets. To resolve this, in this work we also explicitly initialize the translations by fixing the rotation initialization and using

distributed Gauss-Seidel to solve the reduced linear least squares over translations. On the other hand, to prevent significant communication usage in the initialization stage, we limit the number of Gauss-Seidel iterations to 50 for both rotation and translation initialization.

Lastly, we note that both initialization approaches return an initial solution $T \in \mathrm{SE}(d)^n$ on the original search space of PGO. Nevertheless, recall from Algorithm 4.1 that the Riemannian Staircase requires a initial point $X \in \mathcal{M}_{\text{PGO}}(r_0, n)$ on the search space of the rank-restricted SDP (Problem 4.3). We thus need a mechanism that lifts the initial solution from $\mathrm{SE}(d)^n$ to the higher dimensional $\mathcal{M}_{\text{PGO}}(r_0, n)$. This can be achieved by sampling a random point $Y_{\text{rand}} \in \mathrm{St}(d, r)$, and subsequently setting $X = Y_{\text{rand}} T$.

4.6.2 Distributed Rounding

After solving the SDP relaxation, we need to "round" the low-rank factor $X^\star \in \mathcal{M}_{\text{PGO}}(r, n)$ returned by the Riemannian Staircase to a feasible solution to the original PGO problem (see line 4 in Algorithm 4.2). In this section, we describe a distributed rounding procedure that incurs minimal computation and communication costs, and furthermore is guaranteed to return a global minimizer to the original PGO (Problem 2.2) provided that the SDP relaxation is *exact*.

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

$$X^\star = \begin{bmatrix} Y_1^\star & p_1^\star & \dots & Y_n^\star & p_n^\star \end{bmatrix} \in (\mathrm{St}(d, r) \times \mathbb{R}^r)^n. \tag{4.56}$$

In Theorem 4.2, we have shown that if the SDP relaxation is exact, then the first block-row of the corresponding SDP solution, which can be written as $T^\star \triangleq (Y_1^\star)^\top X^\star$, gives a global minimizer to PGO (Problem 2.2). Looking at the rotation and translation of each pose in $T^\star$ separately,

$$R_i^\star = (Y_1^\star)^\top Y_i, \; t_i^\star = (Y_1^\star)^\top p_i. \tag{4.57}$$

Equation (4.57) thus recovers globally optimal rotation and translation estimates. If the SDP relaxation is not exact, the $R_i^\star$ as computed in (4.57) may not be a valid rotation. To ensure feasibility, we additionally project it to $\mathrm{SO}(d)$,

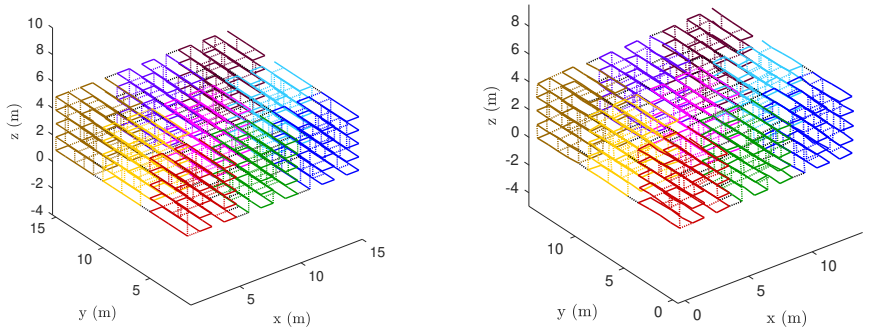
$$R_i = \mathrm{Proj}_{\mathrm{SO}(d)}({Y_1^\star}^\top Y_i^\star). \tag{4.58}$$

In (4.58), the projection can be carried out by computing the SVD.

Remark 4.15 (Communication requirements of distributed initialization and rounding)**.** The distributed chordal initialization [38] has a similar communication pattern as distributed local search, where at each iteration robots exchange messages corresponding to their public poses. On the other hand, distributed rounding incurs minimal communication, since agents only need to relay the small r-by-d matrix $Y_1^\star$ over the network.

4.7 Experiments

We perform extensive evaluations of the proposed DC2-PGO algorithm on both simulations and benchmark CSLAM datatsets. Our simulation consists of multiple robots moving next to each other in a 3D grid with lawn mower trajectories. With a given probability (default 0.3), loop closures are added to connect neighboring poses. For all relative measurements, we simulate isotropic Langevin rotation noise according to (2.19a) with mode I_d and concentration parameter κ. To make the process of setting κ more intuitive, we first set a desired standard deviation σ_R for the *rotation angle* of the rotational noise, and then use the asymptotic approximation shown in SE-Sync [14, Appendix A] to compute the corresponding concentration parameter κ. We also simulate Gaussian translation noise according to (2.19b) with zero mean and standard deviation σ_t. The default noise parameters are $\sigma_R = 3°, \sigma_t = 0.05m$. See Figure 4-3 for an example simulation together with the certified global minimizer found by DC2-PGO (Algorithm 4.2). All implementations are written in MATLAB. All experiments are carried out on a laptop with an Intel i7 CPU and 8 GB RAM.



(a) Ground truth

(b) Certified solution from Algorithm 4.2

Figure 4-3: Example simulation consisting of 9 robots (trajectories shown in different colors) where each robot has 125 poses. Loop closures are drawn as dotted lines. (a) Ground truth. (b) Certified global minimizer returned by DC2-PGO (Algorithm 4.2).

When evaluating our approach and baseline methods, we use the following performance metrics. First, we compute the optimality gap $f - f^\star_{\text{SDP}}$, where $f^\star_{\text{SDP}}$ is the optimal value of the centralized SDP relaxation computed using SE-Sync [14]. In the (typical) case that the SDP relaxation is exact, the rounded PGO estimates returned by both our approach and SE-Sync will achieve a zero optimality gap (to within numerical tolerances). Additionally, when evaluating convergence rates of local search methods, we compute the evolution of the Riemannian gradient norm, which quantifies how fast the iterates are converging to a first-order critical point. Lastly, in some experiments, we also compute the translational and rotational root mean square errors (RMSE) with respect to the solution of SE-Sync. Given two sets of translations $t, t' \in \mathbb{R}^{d\times n}$, the translational RMSE is based on the standard ℓ_2 distance after aligning t, t' in the same frame. Given two sets of rotations $R, R' \in \text{SO}(d)^n$, we define the rotational RMSE to be,

$$\text{RMSE}(R, R') = \sqrt{d_S(R, R')^2/n}, \tag{4.59}$$

where $d_S(R, R')$ is the *orbit distance* for $\text{SO}(d)^n$ defined in SE-Sync [14, Appendix C.1]. Although we measure rotational and translational errors separately, both still give meaningful metrics of the overall PGO solution. This is especially true for

rotational RMSE, since we know that given rotation estimates, the corresponding optimal translations can be computed in closed-form [14].

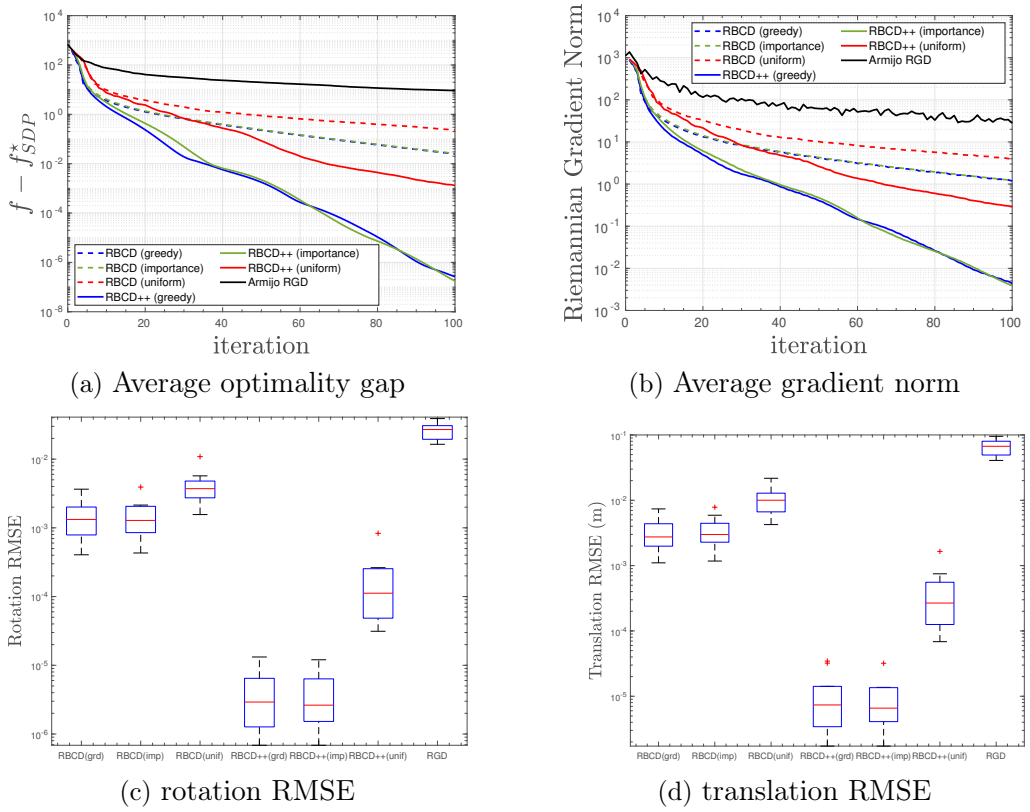
The rest of this section is organized as follows. In Section 4.7.1, we evaluate our distributed local search methods, specifically RBCD and its accelerated version RBCD++, when solving the rank-restricted relaxations of PGO. Then, in Section 4.7.2, we evaluate the proposed distributed verification scheme. Lastly, in Section 4.7.3, we evaluate our complete distributed certifiably correct PGO algorithm (Algorithm 4.2).

4.7.1 Evaluations of Distributed Local Search

We first evaluate the performance of the proposed RBCD and RBCD++ algorithms when solving the rank-restricted relaxations (Problem 4.3). Recall that this serves as the central local search step in our overall Riemannian Staircase algorithm. By default, we set the relaxation rank to $r = 5$, and enable parallel execution as described in Section 4.3.4.

Figure 4-4 shows the performance on our 9 robot scenario shown in Figure 4-3. We report the performance of our proposed methods using all three block selection rules proposed in Section 4.3.1: uniform sampling, importance sampling, and greedy selection. For reference, we also compare our performance against the Riemannian gradient descent (RGD) algorithm with Armijo's backtracking line search implemented in Manopt [181]. As the results demonstrate, both RBCD and RBCD++ dominate the baseline RGD algorithm in terms of convergence speed and solution quality. As expected, importance sampling and greedy block selection also lead to faster convergence compared to uniform sampling. Furthermore, RBCD++ shows significant empirical acceleration and is able to converge to the global minimum with high precision using only 100 iterations. Note that in this experiment, we choose to report convergence speed with respect to iteration number, because it is directly linked to the number of communication rounds required during distributed optimization. For completeness, we also note that the average runtime of BlockUpdate (implemented based on a modified version of the trust-region solver in Manopt [181]) is 0.023 s.

In Figure 4-4, we report the performance of RBCD++ with the default adaptive



(a) Average optimality gap

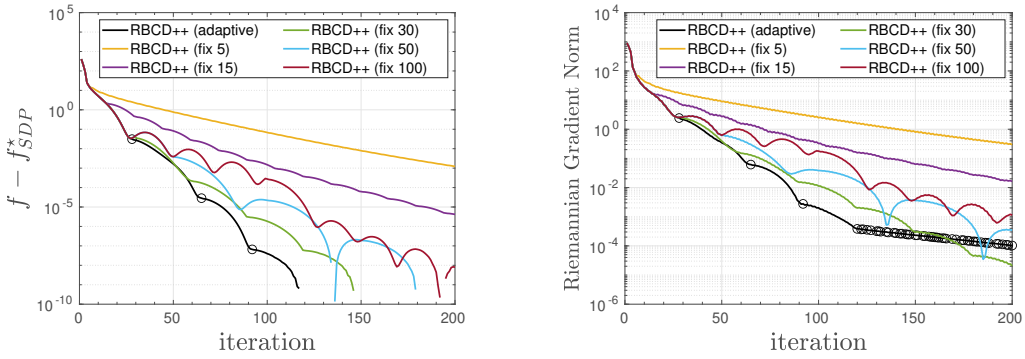
(b) Average gradient norm

(c) rotation RMSE

(d) translation RMSE

Figure 4-4: Convergence rates and final estimation errors of RBCD and RBCD++ with uniform, importance, or greedy selection rules, on the 9 robot simulation shown in Figure 4-3. (a) Evolution of optimality gap averaged over 10 random runs. (b) Evolution of Riemannian gradient norm averaged over 10 random runs. (c) Boxplot of final rotation RMSE (after rounding) with respect to the global minimizer. (d) Boxplot of final translation RMSE (after rounding) with respect to the global minimizer.

restart scheme (see Algorithm 4.5). As we have discussed in Remark 4.8, a less expensive and hence more practical restart scheme is simply to reset Nesterov's acceleration after a fixed number of iterations; this scheme is typically referred to as *fixed restart* in the literature. In Figure 4-5, we compare adaptive restart with fixed restart on a random instance of our simulation. For fixed restart, we use different restart frequency ranging from every 5 to every 100 iterations. We observe that with a short restart period (e.g., 5), convergence of RBCD++ is significantly slowed down. This result is expected, as frequent restarting essentially removes the effect of acceleration from the iterations of RBCD++. In the extreme case of restarting at every iteration, the algorithm essentially reduces to RBCD. On the other hand, long restart period (e.g., 100) also has a negative impact, and we observe that the overall convergence displays



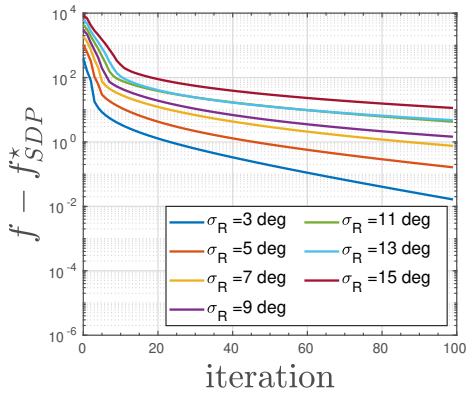
(a) Optimality Gap (b) Gradient norm

Figure 4-5: Adaptive restart vs. fixed restart for RBCD++ (greedy selection) on a random instance of our simulation. For fixed restart, we use restart frequency ranging from every 5 to every 100 iterations. For adaptive restart (black curves), we also highlight iterations where restart is triggered with circle markers. (a) Evolution of optimality gaps. (b) Evolution of Riemannian gradient norm.

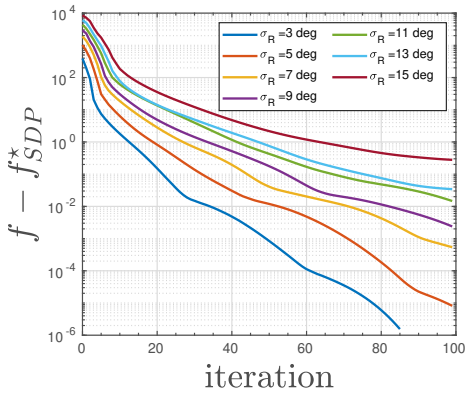
undesirable oscillations. Finally, we find that a suitably chosen restart period (e.g., 30) demonstrates a superior convergence rate that is similar to the adaptive restart scheme.

On the same 9 robot scenario, we also report the convergence of RBCD and RBCD++ (using greedy selection) under increasing measurement noise, shown in Figure 4-6. As expected, as rotation noise increases, the convergence rates of both RBCD and RBCD++ are negatively impacted. On the other hand, we observe that increasing translation noise actually leads to better convergence behavior, as shown in Figure 4-6c-4-6d. To explain these observations, we conjecture that increasing rotation noise and decreasing translation noise magnify the *ill conditioning* of the optimization problem. Qualitatively similar results were also reported in [14] (decreasing translational noise was observed to *increase* SE-Sync's wall-clock time).

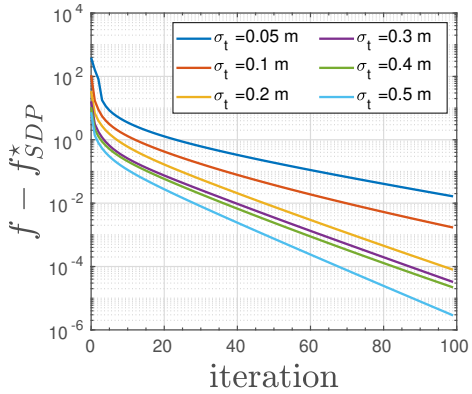
Lastly, we evaluate the scalability of RBCD and RBCD++ (both with greedy block selection) as the number of robots increases from 4 to 49 in the simulation. As each robot has 125 poses, the maximum size of the global PGO problem is 6125. Figure 4-7 reports the convergence speeds measured in Riemannian gradient norm. Both RBCD and RBCD++ are reasonably fast for small number of robots. Nevertheless, the non-accelerated RBCD algorithm begins to show slow convergence as the number of



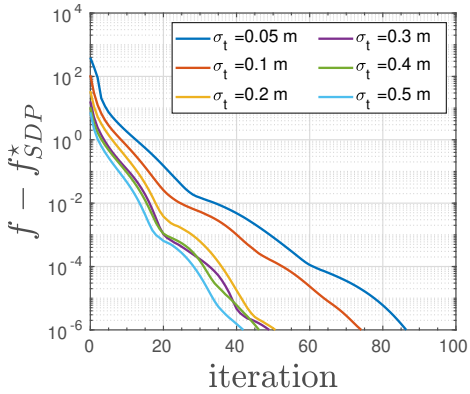
(a) RBCD under increasing rotation noise



(b) RBCD++ under increasing rotation noise



(c) RBCD under increasing translation noise



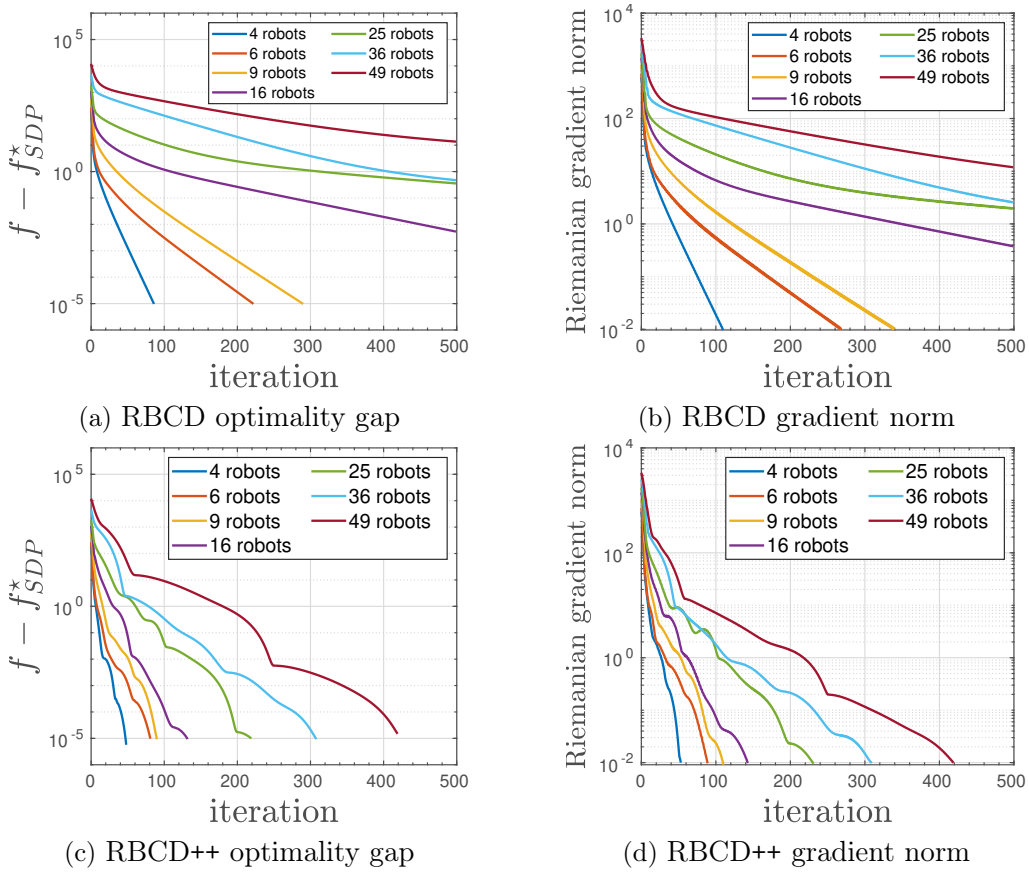
(d) RBCD++ under increasing translation noise

Figure 4-6: Convergence of RBCD and RBCD++ with greedy selection under varying rotation and translation measurement noise. (a)-(b) Convergence of RBCD and RBCD++ under increasing rotation noise and fixed translation noise of $\sigma_t = 0.05$ m. (c)-(d) Convergence of RBCD and RBCD++ under increasing translation noise and fixed rotation noise of $\sigma_t = 3°$. All results are averaged across 10 random runs.

robots exceeds 16. In comparison, our accelerated RBCD++ algorithm shows superior empirical convergence speed, even in the case of 49 robots. We note that in this case although RBCD++ uses 400 iterations to achieve a Riemannian gradient norm of 10^{-2}, the actual optimality gap (Figure 4-7c) decreases much more rapidly to 10^{-5}, which indicates that our solution is very close to the global minimum.

4.7.2 Evaluations of Distributed Verification

In this section we evaluate our proposed distributed verification method. Recall from Section 4.5 that the bulk of work happens when using accelerated power iteration



(a) RBCD optimality gap

(b) RBCD gradient norm

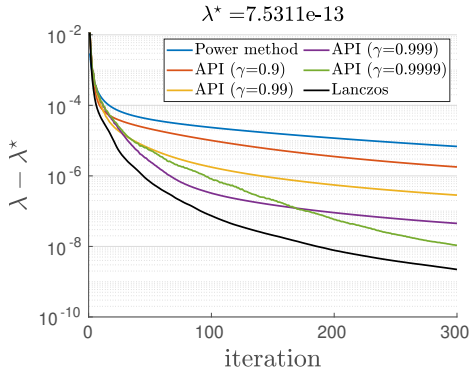
(c) RBCD++ optimality gap

(d) RBCD++ gradient norm

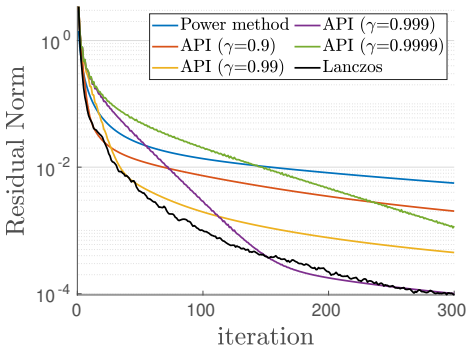
Figure 4-7: Scalability of RBCD and RBCD++ with greedy selection as the number of robots increases. Each robot has 125 poses. Convergence speed is measured in terms of the Riemannian gradient norm.

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

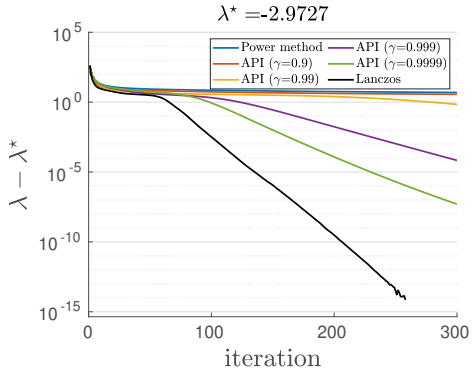
From Remark 4.13, the accelerated power iteration achieves the theoretical optimal rate when the employed momentum term satisfies $\beta \approx \lambda_2^2/4$, where λ_2 is the second dominant eigenvalue of C. Since we know that λ_{dom} belongs to a tight cluster whenever X is globally optimal, we expect that typically $\lambda_2 \approx \lambda_{\text{dom}}$. Using this insight, in our experiment we first estimate λ_2 by multiplying λ_{dom} with a factor $\gamma < 1$ that is close to one, i.e., $\widehat{\lambda}_2 = \gamma \lambda_{\text{dom}}$. Subsequently we use this estimated value to set



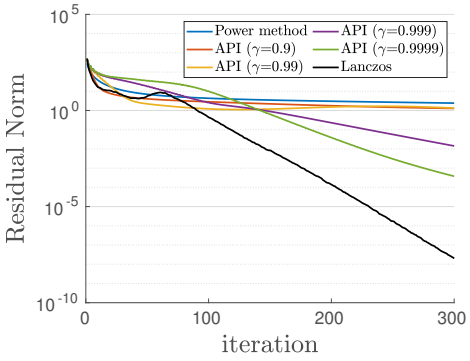
(a) Distance to minimum eigenvalue when verifying global minimizer



(b) Eigenvector residual norm when verifying global minimizer



(c) Distance to minimum eigenvalue when verifying suboptimal local minimizer



(d) Eigenvector residual norm when verifying suboptimal local minimizer

Figure 4-8: Performance of accelerated power iteration (API) on the `Killian court` dataset, using different values of γ to set the momentum term according to (4.60). (a)-(b) Verification of a global minimizer computed by SE-Sync. (c)-(d) Verification of a suboptimal first-order critical point. In both cases, the minimum eigenvalue of the dual certificate matrix (denoted as $\lambda^\star$) is computed using the `eigs` function in MATLAB.

the momentum term,

$$\beta = \widehat{\lambda}_2^2/4 = \gamma^2 \lambda_{\text{dom}}^2/4. \tag{4.60}$$

We design two test cases using the `Killian court` dataset. In the first case, we verify the global minimizer computed by SE-Sync [14]. By Theorem 4.6, the dual certificate matrix $S(X)$ must be positive semidefinite. Furthermore, since $S(X)$ always has a nontrivial nullspace spanned by the rows of the corresponding primal solution, we expect the minimum eigenvalue of $S(X)$ to be zero in this case. Indeed, when computing this using the `eigs` function in MATLAB, the final value (denoted as $\lambda^\star$ in Figure 4-8a) is close to zero to machine precision. Figure 4-8a shows how fast

each method converges to $\lambda^\star$, where we use an initial eigenvector estimate obtained by (slightly) randomly perturbing a row of the global minimizer [168]. Figure 4-8b shows the corresponding Ritz residual for the estimated eigenvector v. Assuming v is normalized, this is given by,

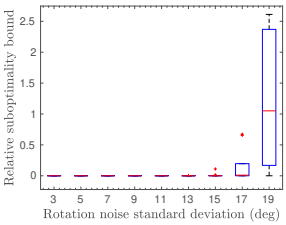
$$\text{ResidualNorm}(v) = \left\| S(X)v - (v^\top S(X)v)v \right\|_2 . \tag{4.61}$$

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

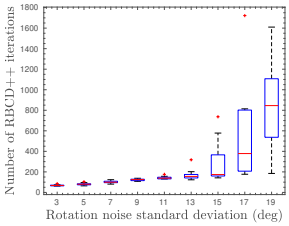
In the second case, we verify a suboptimal first-order critical point obtained by running RBCD++ with $r = d$ using random initialization. We verify that the minimum eigenvalue of $S(X)$ is negative (≈ -2.97), which is consistent with the prediction of Theorem 4.6. In this case, we observe that using a *random* initial eigenvector estimate leads to better convergence compared to obtaining the initial estimate from a perturbed row of the primal solution. Intuitively, using the perturbed initial guess would cause the iterates of power method to be "trapped" for longer period of time near the zero eigenspace spanned by the rows of X. Figure 4-8c-4-8d shows results generated with random initial eigenvector estimate. Note that there is a bigger performance gap between accelerated power iteration and the Lanczos algorithm. However, we also note that in reality, full convergence is actually not needed in this case. Indeed, from Theorem 4.6, we need only identify *some* direction that satisfies $v^\top S(X)v < 0$ in order to escape the current suboptimal solution.

4.7.3 Evaluations of Complete Algorithm (Algorithm 4.2)

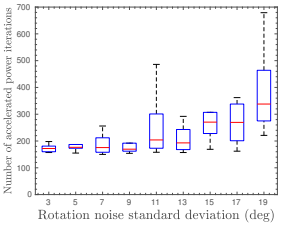
So far, we have separately evaluated the proposed local search and verification techniques. In this section, we evaluate the performance of the complete DC2-PGO algorithm (Algorithm 4.2) that uses distributed Riemannian Staircase (Algorithm 4.1) to solve the SDP relaxation of PGO. By default, at each level of the Staircase we use RBCD++ with greedy selection to solve the rank-restricted relaxation until the



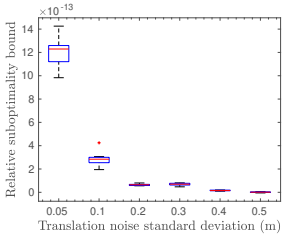
(a) Relative suboptimality bound under increasing rotation noise



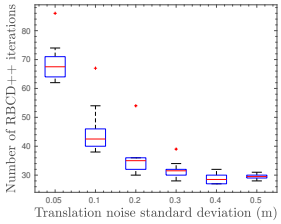
(b) Total number of RBCD++ iterations under increasing rotation noise



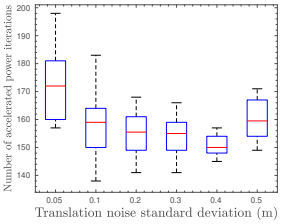
(c) Total number of accelerated power iterations under increasing rotation noise



(d) Relative suboptimality bound under increasing translation noise



(e) Total number of RBCD++ iterations under increasing translation noise



(f) Total number of accelerated power iterations under increasing translation noise

Figure 4-9: Evaluation of the proposed DC2-PGO (Algorithm 4.2) under increasing measurement noise. At each noise level, we simulate 10 random realizations of the 9-robot scenario shown in Figure 4-3. Left column shows boxplot of relative suboptimality bound $(f(T) - f^\star_{\text{SDP}})/f^\star_{\text{SDP}}$. Middle and right columns show boxplots of total number of iterations used by RBCD++ and accelerated power method. The top row shows results under increasing rotation noise $\sigma_R \in [3, 19]$ deg and fixed translation noise $\sigma_t = 0.05$ m. The bottom row shows results under increasing translation noise $\sigma_t \in [0.05, 0.5]$ m and fixed rotation noise $\sigma_R = 3$ deg.

Riemannian gradient norm reaches 10^{-1}. Then, we use the accelerated power method to verify the obtained solution. To set the momentum term, we employ the same method introduced in the last section with $\gamma = 0.999$ in (4.60). The accelerated power iteration is deemed converged when the eigenvector residual defined in (4.61) reaches 10^{-2}.

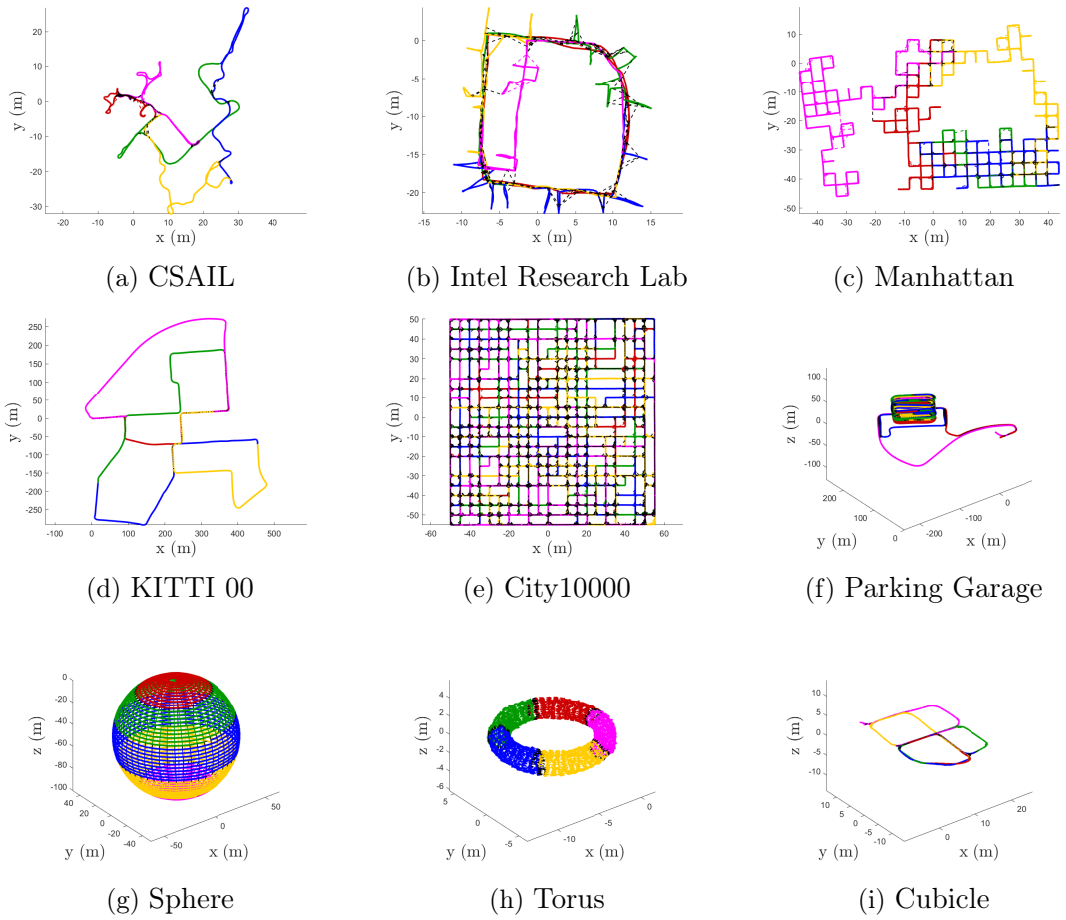
We first examine the exactness of the SDP relaxation in the 9-robot scenario shown in Figure 4-3 under increasing measurement noise. Recall that DC2-PGO returns both a rounded feasible solution $T \in \text{SE}(d)^n$ as well as the optimal value of the SDP relaxation $f^\star_{\text{SDP}}$. To evaluate exactness, we record the upper bound on the relative suboptimality of T, defined as $(f(T) - f^\star_{\text{SDP}})/f^\star_{\text{SDP}}$. We note that a zero

Table 4.1: Evaluation on benchmark PGO datasets. Each dataset simulates a CSLAM scenario with five robots. On each dataset, we report the objective value achieved by initialization, centralized SE-Sync [14], the distributed Gauss-Seidel (DGS) with SOR parameter 1.0 as recommended in [38], and the proposed DC2-PGO algorithm. For the latter two distributed algorithms, we also report the total number of local search iterations. On each dataset, we highlight the distributed algorithm that (i) achieves lower objective and (ii) uses less local search iterations. On all datasets, our approach is able to verify its solution as the global minimizer. We note that the numerical difference with SE-Sync on some datasets is due to the looser convergence condition during distributed local search.

Dataset	# Vertices	# Edges	Objective				Local Search Iterations	
			Init.	SE-Sync [14]	DGS [38]	DC2-PGO	DGS [38]	DC2-PGO
Killian Court (2D)	808	827	229.0	61.15	63.52	**61.22**	27105	**189**
CSAIL (2D)	1045	1171	31.50	31.47	31.49	**31.47**	**10**	197
Intel Research Lab (2D)	1228	1483	396.6	393.7	428.89	**393.7**	**10**	187
Manhattan (2D)	3500	5453	369.0	193.9	242.05	**194.0**	1585	**785**
KITTI 00 (2D)	4541	4676	1194	125.7	269.87	**125.7**	**1485**	2750
City10000 (2D)	10000	20687	5395	638.6	2975.2	**638.7**	2465	**1646**
Parking Garage (3D)	1661	6275	1.64	1.263	1.33	**1.311**	**25**	47
Sphere (3D)	2500	4949	1892	1687	1689	**1687**	300	**53**
Torus (3D)	5000	9048	24617	24227	24246	**24227**	100	**88**
Cubicle (3D)	5750	16869	786.0	717.1	726.69	**717.1**	**45**	556
Rim (3D)	10195	29743	8177	5461	5960.4	**5461**	**515**	1563

suboptimality bound means that the SDP relaxation is exact and the solution T is a global minimizer. As shown in the first column of Figure 4-9, DC2-PGO is capable of finding global minimizers for all translation noise considered in our experiments, and for rotation noise up to 11 degree, which is still much larger than noise magnitude typically encountered in SLAM. The middle and right columns of Figure 4-9 show the total number of iterations used by RBCD++ and accelerated power method, across all levels of the staircase. Interestingly, we observe that changing measurement noise has a greater impact for distributed local search compared to distributed verification.

Lastly, we evaluate DC2-PGO on benchmark datasets. Figure 4-10 shows the globally optimal solutions returned by our algorithm. In Table 4.1, we compare the performance of DC2-PGO against the centralized certifiable SE-Sync algorithm [14], as well as the state-of-the-art distributed Gauss-Seidel (DGS) algorithm by Choudhary et al. [38]. For DGS, we set the SOR parameter to 1.0 as recommended by the authors, and for which we also observe stable performance in general. On all datasets, DC2-PGO is able to verify its solution as the global minimizer. We note that on some



(a) CSAIL (b) Intel Research Lab (c) Manhattan

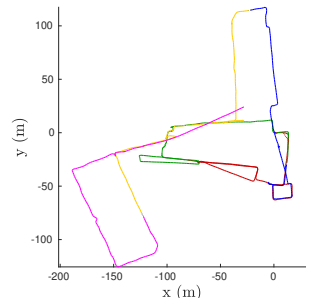
(d) KITTI 00 (e) City10000 (f) Parking Garage

(g) Sphere (h) Torus (i) Cubicle

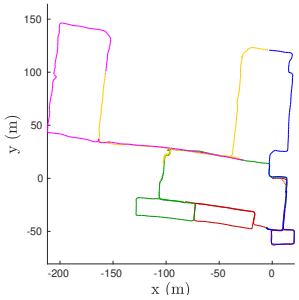
Figure 4-10: Globally optimal estimates returned by DC2-PGO (Algorithm 4.2) on benchmark datasets.

datasets, the final objective value is slightly higher than SE-Sync. This is due to the looser convergence condition used in our distributed local search: for RBCD++ we set the gradient norm threshold to 10^{-1}, while for SE-Sync we set the threshold to 10^{-6} in order to obtain a high-precision reference solution.[8] On the other hand, DC2-PGO is clearly more advantageous compared to DGS, as it returns a global minimum, often with fewer iterations. The performance of DGS is more sensitive to the quality of initialization, as manifested on the `Killian Court` and `City10000` datasets. In addition, while our local search methods are guaranteed to reduce the objective value at each iteration, DGS does *not* have this guarantee as it is operating on a linearized

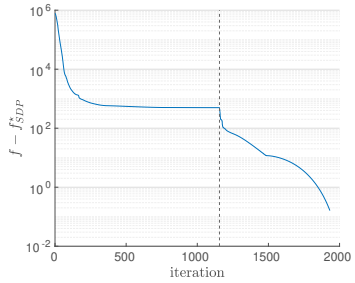
[8] In general it is not reasonable to expect RBCD or RBCD++ to produce solutions that are as precise as those achievable by SE-Sync in tractable time, since the former are *first-order* methods, while the latter is *second-order*.



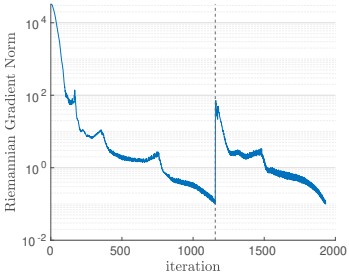
(a) Suboptimal critical point at rank $r = 3$



(b) Certified global minimizer at rank $r = 4$



(c) Combined optimality gap ($r = 3$ and $r = 4$) during local search



(d) Combined Riemannian gradient norm ($r = 3$ and $r = 4$) during local search

Figure 4-11: DC2-PGO (Algorithm 4.2) returns the global minimizer of the `Killian court` dataset even from random initialization. (a) From random initialization, RBCD++ converges to a suboptimal critical point at rank $r = 3$. (b) Via distributed verification and saddle point escaping, our algorithm is able to escape the suboptimal solution and converges to the global minimizer at rank $r = 4$. (c) Evolution of optimality gap across $r = 3$ and $r = 4$. (d) Evolution of Riemannian gradient norm across $r = 3$ and $r = 4$. The vertical dashed line indicates the transition from $r = 3$ to $r = 4$.

approximation of the PGO problem.

To further demonstrate the uniqueness of our algorithm as a *global* solver, we show that it is able to converge to the global minimum even from random initialization. This is illustrated using the `Killian court` dataset in Figure 4-11. Due to the random initialization, the first round of distributed local search at rank $r = 3$ converges to a suboptimal critical point. This can be seen in Figure 4-11c, the optimality gap at $r = 3$ (first 1068 iterations) converges to a non-zero value. From distributed verification, our algorithm detects the solution as a saddle point and is able to escape and converges to the correct global minimizer at rank $r = 4$. This can be clearly seen in Figure 4-11d, where escaping successfully moves the iterate to a position with

large gradient norm, from where local search can successfully descend to the global minimizer.

4.8 Conclusion

This chapter developed the first *certifiably correct* algorithm for *distributed* pose graph optimization. Our method is based upon a sparse semidefinite relaxation of the PGO problem that we prove enjoys the same exactness guarantees as current state-of-the-art centralized methods [14]: namely, that its minimizers are *low-rank* and provide *globally optimal solutions* of the original PGO problem under moderate noise. To solve large-scale instances of this relaxation in the distributed setting, we leveraged the existence of low-rank solutions to propose a distributed Riemannian Staircase framework, employing Riemannian block coordinate descent as the core distributed optimization method. We proved that RBCD enjoys a *global* sublinear convergence rate under standard (mild) conditions, and can be significantly accelerated using Nesterov's scheme. We also developed the first distributed solution verification and saddle escape algorithms to *certify* the optimality of critical points recovered via RBCD, and to descend from suboptimal critical points if necessary. Finally, we provided extensive numerical evaluations, demonstrating that the proposed approach correctly recovers globally optimal solutions under moderate noise, and outperforms alternative distributed methods in terms of estimation quality and convergence speed.

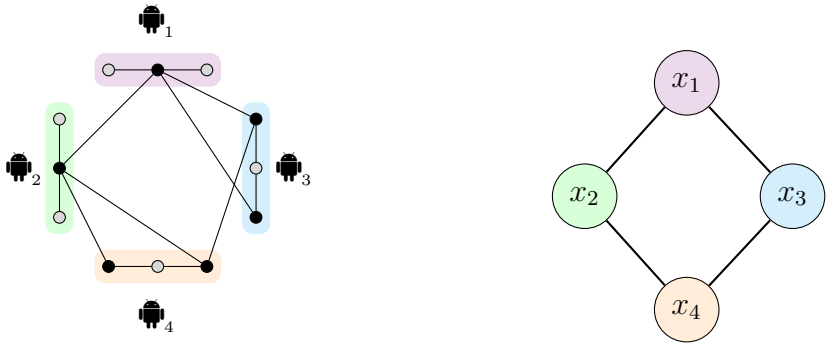
Chapter 5

Asynchronous Distributed Pose Graph Optimization

5.1 Introduction

In Chapter 4, we developed distributed local search procedures for PGO and its rank-restricted relaxations. However, both RBCD and RBCD++ developed in Chapter 4 are inherently *synchronous*, which necessitates that robots, for instance, pass messages over the network or wait at predetermined points, in order to ensure up-to-date information sharing during distributed optimization. In practice, doing so may incur considerable communication overhead and increase the complexity of implementation.

In this chapter, we complement the algorithmic contributions of Chapter 4 by developing an *asynchronous* algorithm for distributed PGO that we call ASAPP (**A**synchronous **S**toch**A**stic **P**arallel **P**ose Graph Optimization). We take inspiration from existing parallel and asynchronous algorithms [58, 155–157, 159], and adapt these ideas to solve the *non-convex* Riemannian optimization problem underlying PGO. In ASAPP, each robot executes its local optimization loop at a high rate, without waiting for updates from others over the network. This makes ASAPP easier to implement in practice and flexible against communication delay. Furthermore, we show that ASAPP can support both the maximum likelihood estimation (MLE) formulation of PGO (Problem 2.2) and its rank-restricted relaxations (Problem 4.3)



(a) Pose graph G (b) Robot-level graph $G_{\mathcal{R}}$

Figure 5-1: (a) Example pose graph G with 4 robots, each with 3 poses. Each edge denotes a relative pose measurement. Private poses are colored in gray. (b) Corresponding robot-level graph $G_{\mathcal{R}}$. Two robots are connected if they share any relative measurements (inter-robot loop closures). Note that at any time during distributed optimization, robots do not need to share their private poses with any other robots.

studied in Chapter 4.

Contributions. Since asynchronous algorithms allow communication delays to be substantial and unpredictable, it is usually unclear under what conditions they converge in practice. In this chapter, we provide a rigorous answer to this question and establish the first known convergence result for asynchronous algorithms on the *non-convex* PGO problem. In particular, we show that as long as the worst-case delay is not arbitrarily large, ASAPP with a sufficiently small stepsize always converges to first-order critical points when solving PGO and its rank-restricted relaxations, with *global* sublinear convergence rate. The derived stepsize depends on the maximum delay and inherent problem sparsity, and furthermore reduces to the well known constant of $1/L$ (where L is the Lipschitz constant) for synchronous algorithms when there is no delay. Numerical evaluations on simulated and real-world datasets demonstrate that ASAPP compares favorably against state-of-the-art synchronous methods, and furthermore is resilient against a wide range of communication delays. Both results show the practical value of the proposed algorithm in a realistic distributed setting.

5.2 Problem Formulation

In this chapter, we use $\mathcal{R} = \{1, 2, \ldots, n\}$ to denote a set of n robots, and $i, j \in \mathcal{R}$ serve as indices that refer to specific robots. We consider solving the MLE formulation of collaborative PGO (Problem 2.2), as well as its rank-restricted relaxations (Problem 4.3) studied in Chapter 4. For the purpose of designing decentralized algorithms (Section 5.3), it is more convenient to rewrite both problems into a more abstract form at the level of robots, which has the following form.

Problem 5.1 (Robot-level Optimization Problem)**.**

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E_{\mathcal{R}}} f_{ij}(x_i, x_j) + \sum_{i\in\mathcal{R}} h_i(x_i), \\ \text{s.t.} \quad & x_i \in \mathcal{M}_i, \ \forall i \in \mathcal{R}. \end{aligned} \tag{5.1}$$

In (5.1), each variable x_i concatenates all variables owned by robot $i \in \mathcal{R}$. For instance, in the case of solving the rank-restricted relaxations (Problem 4.3), x_i contains all the "lifted" rotation and translation variables of robot i, *i.e.*,

$$x_i = \begin{bmatrix} Y_{i_1} & p_{i_1} & \ldots & Y_{i_{n_i}} & p_{i_{n_i}} \end{bmatrix}, \tag{5.2}$$

$$\mathcal{M}_i = (\mathrm{St}(d, r) \times \mathbb{R}^r)^{n_i}, \tag{5.3}$$

where $Y_{i_k} \in \mathrm{St}(d, r)$ and $p_{i_k} \in \mathbb{R}^p$ correspond to the k-th "lifted" rotation and translation of robot i, and n_i is the total number of pose variables owned by robot i.

The cost function in (5.1) consists of a set of *shared costs* $f_{ij} : \mathcal{M}_i \times \mathcal{M}_j \to \mathbb{R}$ between pairs of robots, and a set of *private costs* $h_i : \mathcal{M}_i \to \mathbb{R}$ for individual robots. Intuitively, f_{ij} is formed by relative measurements between any of robot i's poses and j's poses (*i.e.*, inter-robot loop closures). In contrast, h_i is formed by relative measurements within robot i's own trajectory. The sparsity structure of (5.1), specified by the robot-level edge set $E_{\mathcal{R}}$, corresponds exactly to the robot-level dependency graph $G_{\mathcal{R}} \triangleq (\mathcal{R}, E_{\mathcal{R}})$ first introduced in Section 2.2.2. For convenience, Figure 5-1 shows an example of pose graph together with the corresponding robot-

level dependency graph $G_{\mathcal{R}}$. Recall that each vertex in $G_{\mathcal{R}}$ corresponds to the entire trajectory of a single robot $i \in \mathcal{R}$. Two robots i, j are connected in $G_{\mathcal{R}}$ if they share any inter-robot loop closures. In this case, we call j a *neighboring robot* of i, and the poses that have inter-robot loop closures *neighboring poses*. If a pose variable is not a neighboring pose to any other robots (*i.e.*, it does not have any inter-robot loop closure), we call this pose a *private pose*. For robot i to evaluate the shared cost f_{ij}, it only needs to know its neighboring poses in robot j's trajectory (see Figure 5-1). This property is crucial in preserving the *privacy* of participating robots, because at any time a robot does not need to share its private poses with any of its teammates.

5.3 Proposed Algorithm

We present our main algorithm, Asynchronous Stochastic Parallel Pose Graph Optimization (ASAPP), for solving distributed PGO problems of the form (5.1). Our algorithm is inspired by asynchronous stochastic coordinate descent (*e.g.*, see [157]). In the context of distributed PGO, each coordinate corresponds to the stacked pose variables x_i of a single robot as defined in (5.1).

In a practical multi-robot SLAM scenario, each robot can optimize its own pose estimates at any time, and can additionally share its (non-private) poses with others when communication is available. Correspondingly, each robot running ASAPP has two concurrent onboard processes, which we refer to as the *optimization thread* and *communication thread*. We emphasize that the robots perform both optimization and communication completely in parallel and without synchronization with each other. We begin by describing the communication thread and then proceed to the optimization thread. Without loss of generality, we describe the algorithm from the perspective of robot $i \in \mathcal{R}$.

5.3.1 Communication Thread

As part of the communication module, each robot $i \in \mathcal{R}$ implements a local data structure, called a *cache*, that contains the robot's own variable x_i, together with

the most recent copies of neighboring poses received from the robot's neighbors. A very similar design that allows asynchronous communication is proposed by Cunningham *et al.* [55–57], although the authors have not discussed convergence in the asynchronous setting.

Since only i can modify x_i, the value of x_i in robot i's cache is guaranteed to be up-to-date at anytime. In contrast, the copies of neighboring poses from other robots can be *out-of-date* due to communication delay. For example, by the time robot i receives and uses a copy of robot j's poses, j might have already updated its poses due to its local optimization process. In Section 5.4, we show that ASAPP is resilient against such network delay. Nevertheless, for ASAPP to converge, we still assume that the total delay induced by the communication process remains *bounded* (Section 5.4). The communication thread performs the following two operations over the cache.

• **Receive**: After receiving a neighboring pose from a neighboring robot j over the network, the communication thread updates the corresponding entry in the cache to store the new value.

• **Send**: Periodically (when communication is available), robot i also transmits its latest public pose variables (*i.e.*, poses that have inter-robot measurements with other robots) to its neighbors. Recall from Section 5.2 that robot i does not need to send its private poses, as these poses are not needed by other robots to optimize their estimates.

5.3.2 Optimization Thread

Concurrent to the communication thread, the optimization thread is invoked by a local clock that ticks according to a Poisson process of rate $\lambda > 0$.

Definition 5.1 (Poisson process [182])**.** Consider a sequence $\{X_1, X_2, ...\}$ of positive, independent random variables that represent the time elapsed between consecutive events (in this case, clock ticks). Let $N(t)$ be the number of events up to time $t \geq 0$. The counting process $\{N(t), t \geq 0\}$ is a Poisson process with rate $\lambda > 0$ if the

interarrival times $\{X_1, X_2, ...\}$ have a common exponential distribution function,

$$P(X_k \leq a) = 1 - e^{-\lambda a},\ a \geq 0. \tag{5.4}$$

The use of Poisson clocks originates from the design of randomized gossip algorithms by Boyd *et al.* [183] and is a commonly used tool for analyzing the global behavior of distributed randomized algorithms. The rate parameter λ is equal among robots. In practice, we can adjust λ based on the extent of network delay and the robots' computational capacity. Using this local clock, the optimization thread performs the following operations in a loop.

• **Read**: For each neighboring robot $j \in \mathrm{Nbr}(i)$, read the value of x_j stored in the local cache. Denote the read values as $\hat{x}_j$. Recall that $\hat{x}_j$ can be *outdated*, for example if robot i has not received the latest messages from robot j. In addition, read the value of x_i, denoted as $\hat{x}_i$. Recall from Section 5.3.1 that $\hat{x}_i$ is guaranteed to be up-to-date.

In practice, $\hat{x}_j$ only contains the set of neighboring poses from robot j since f_{ij} is independent from the rest of j's poses (Fig. 5-1). However, for ease of notation and analysis, we treat $\hat{x}_j$ as if it contains the entire set of j's poses.

• **Compute**: Form the local cost function for robot i, denoted as $g_i(x_i) : \mathcal{M}_i \to \mathbb{R}$, by aggregating relevant costs in (5.1) that involve x_i,

$$g_i(x_i) = h_i(x_i) + \sum_{j \in \mathrm{Nbr}(i)} f_{ij}(x_i, \hat{x}_j). \tag{5.5}$$

Compute the Riemannian gradient at robot i's current estimate $\hat{x}_i$,

$$\eta_i = \mathrm{grad}\, g_i(\hat{x}_i) \in T_{\hat{x}_i}\mathcal{M}_i. \tag{5.6}$$

• **Update**: At the next local clock tick, update x_i in the direction of the negative gradient,

$$x_i \leftarrow \mathrm{Retr}_{\hat{x}_i}(-\gamma \eta_i), \tag{5.7}$$

where $\gamma > 0$ is a constant stepsize. Equation (5.7) gives the simplest update rule that robots can follow. In Section 5.4, we further prove convergence for this update rule.

To accelerate convergence in practice, SE-Sync [14] and Cartan-Sync [20] use a heuristic known as *preconditioning*, which is also applicable to ASAPP. With preconditioning, the following alternative update direction is used,

$$x_i \leftarrow \mathrm{Retr}_{\hat{x}_i}(-\gamma \operatorname{Precon} g_i(\hat{x}_i)[\eta_i]). \tag{5.8}$$

In (5.8), $\operatorname{Precon} g_i(\hat{x}_i) : T_{\hat{x}_i}\mathcal{M}_i \to T_{\hat{x}_i}\mathcal{M}_i$ is a linear, symmetric, and positive definite mapping on the tangent space that approximates the inverse of Riemannian Hessian. Intuitively, preconditioning helps first-order methods benefit from using the (approximate) second-order geometry of the cost function, which often results in significant speedup especially on poorly conditioned problems.

5.3.3 Implementation Details

To make the local clock model valid, we require that the total execution time of the **Read-Compute-Update** sequence be smaller than the interarrival time of the Poisson clock, so that the current sequence can finish before the next one starts. This requirement is fairly lax in practice, as all three steps only involve minimal computation and access to local memory. In the worst case, since the interarrival time is determined by $1/\lambda$ on average [182], one can also decrease the clock rate λ to create more time for each update.

In addition, we note that although the optimization and communication threads run concurrently, minimal thread-level synchronization is required to ensure the so-called *atomic read and write* of individual poses. Specifically, a thread cannot read a pose in the cache if the other thread is actively modifying its value (otherwise the read value would not be valid). Such synchronization can be easily enforced using software locks.

Algorithm 5.1 GLOBAL VIEW OF ASAPP (For Analysis Only)

Input:

Initial solution $x^0 \in \mathcal{M}$ and stepsize $\gamma > 0$.

1: **for** global iteration $k = 0, 1, \ldots$ **do**
2: Select robot $i_k \in \mathcal{R}$ uniformly at random.
3: Read $\hat{x}_{i_k} = x^k_{i_k}$.
4: Read $\hat{x}_{j_k} = x^{k-B(j_k)}_{j_k}$, $\forall j_k \in \mathrm{Nbr}(i_k)$.
5: Compute local gradient $\eta^k_{i_k} = \mathrm{grad}\, g_{i_k}(\hat{x}_{i_k})$.
6: Update $x^{k+1}_{i_k} = \mathrm{Retr}_{\hat{x}_{i_k}}(-\gamma \eta^k_{i_k})$.
7: Carry over all $x^{k+1}_j = x^k_j$, $\forall j \neq i_k$.
8: **end for**

5.4 Convergence Analysis

5.4.1 Global View of the Algorithm

In Section 5.3, we described ASAPP from the local perspective of each robot. For the purpose of establishing convergence, however, we need to analyze the systematic behavior of this algorithm from a global perspective [157, 158, 160, 183]. To do so, let $k = 0, 1, \ldots$ be a virtual counter that counts the total number of **Update** operations applied by all robots. In addition, let the random variable $i_k \in \mathcal{R}$ represent the robot that updates at global iteration k. We emphasize that k and i_k are purely used for theoretical analysis, and are unknown to any of the robots in practice.

Recall from Section 5.3.2 that all **Update** steps are generated by $n = |\mathcal{R}|$ independent Poisson processes, each with rate λ. In the global perspective, merging these local processes is equivalent to creating a single, global Poisson clock with rate λn. Furthermore, at any time, all robots have equal probabilities of generating the next **Update** step, *i.e.*, for all $k \in \mathbb{N}$, i_k is i.i.d. uniformly distributed over the set $\mathcal{R}$. See [182] for proofs of these results.

Using this result, we can write the iterations of ASAPP from the global view; see Algorithm 5.1. We use $x^k \triangleq \begin{bmatrix} x^k_1 & x^k_2 & \ldots & x^k_n \end{bmatrix}$ to represent the value of all robots' poses after k global iterations (*i.e.*, after k total **Update** steps). Note that x lives on the product manifold $\mathcal{M} \triangleq \mathcal{M}_1 \times \mathcal{M}_2 \times \ldots \mathcal{M}_n$. At global iteration k, a robot i_k is selected from $\mathcal{R}$ uniformly at random (line 2). Robot i_k then follows the steps

in Section 5.3.2 to update its own variable (line 3-6). We have used the fact that $\hat{x}_{i_k}$ is always up-to-date (line 3), while $\hat{x}_{j_k}$ is outdated for $B(j_k)$ total **Update** steps (line 4). Except robot i_k, all other robots do not update (line 7).

5.4.2 Sufficient Conditions for Convergence

We establish sufficient conditions for ASAPP to converge to first-order critical points. We adopt the commonly used *partially asynchronous* model [58], which assumes that delay caused by asynchrony is not arbitrarily large. In practice, the magnitude of delay is affected by various factors such as the rate of communication (Section 5.3.1), the rate of local optimization (Section 5.3.2), and intrinsic network latency. For the purpose of analysis, we assume that all these factors can be summarized into a single constant B, which bounds the maximum delay in terms of number of *global iterations* (*i.e.*, **Update** steps applied by all robots) in Algorithm 5.1.

Assumption 5.1 (Bounded Delay)**.** In Algorithm 5.1, there exists a constant $B \geq 0$ such that $B(j_k) \leq B$ for all j_k.

Assumption 5.1 imposes a worst-case upper bound on the delay, and allows the actual delay to fluctuate within this upper bound. In addition, for both PGO (Problem 2.2) and its rank-restricted relaxations (Problem 4.3), the gradients enjoy a Lipschitz-type condition, which is proved in Appendix A.2.4 as part of the algorithmic developments in Chapter 4. For convenience, we restate this result in the following lemma.

Lemma 5.1 (Lipschitz-type gradient for pullbacks)**.** *Let $f : \mathcal{M} \to \mathbb{R}$ denote the cost function in PGO (Problem 2.2) or its rank-restricted relaxation (Problem 4.3). Define the* pullback *cost as $\hat{f}_x \triangleq f \circ \mathrm{Retr}_x : T_x\mathcal{M} \to \mathbb{R}$. There exists a constant $L \geq 0$ such that for any $x \in \mathcal{M}$ and $\eta \in T_x\mathcal{M}$,*

$$\left|\widehat{f}_x(\eta) - [f(x) + \langle \eta, \operatorname{grad} f(x)\rangle]\right| \leq \frac{L}{2}\|\eta\|^2 . \tag{5.9}$$

Using the bounded delay assumption and the Lipschitz-type condition in (5.9), we can proceed to analyze the change in cost function after a single iteration of Algorithm 5.1 (in the global view). We formally state the result in the following lemma.

Lemma 5.2 (Descent Property of Algorithm 5.1)**.** *Under Assumption 5.1, each iteration of Algorithm 5.1 satisfies,*

$$\begin{aligned} f(x^{k+1}) - f(x^k) \leq & -\frac{\gamma}{2}\left\|\operatorname{grad}_{i_k} f(x^k)\right\|^2 - \frac{\gamma - L\gamma^2}{2}\left\|\eta_{i_k}^k\right\|^2 \\ & + \frac{\Delta B L^2 \alpha^2 \gamma^3}{2} \sum_{j_k \in Nbr(i_k)} \sum_{k'=k-B}^{k-1} \left\|\eta_{j_k}^{k'}\right\|^2, \end{aligned} \tag{5.10}$$

where η_i^k denotes the update taken by robot i_k, $\operatorname{grad}_{i_k} f(x^k)$ is the component of the Riemannian gradient that corresponds to robot i_k, $\alpha > 0$ is a constant related to the retraction, and $\Delta > 0$ is the maximum degree of the robot-level graph $G_{\mathcal{R}}$.

In (5.10), the last term on the right-hand side sums over the squared norms of a set of $\{\eta_{j_k}^{k'}\}$, where each $\eta_{j_k}^{k'}$ corresponds to the update taken by a neighbor j_k at an earlier iteration k'. This term is a direct consequence of delay in the system, and is also the main obstacle for proving convergence in the asynchronous setting. Indeed, without this term, it is straightforward to verify that any stepsize that satisfies $0 < \gamma < 1/L$ guarantees $f(x^{k+1}) \leq f(x^k)$, and thus leads to convergent behavior. With the last term in (5.10), however, the overall cost could increase after each iteration.

While the delay-dependent error term gives rise to additional challenges, our next theorem states that with sufficiently small stepsize, this error term is inconsequential and ASAPP provably converges to first-order critical points.

Theorem 5.1 (Global convergence of ASAPP)**.** *Let $f^\star$ be any global lower bound on the optimum of* (5.1)*. Define $\rho \triangleq \Delta/n$. Let $\bar{\gamma} > 0$ be an upper bound on the stepsize that satisfies,*

$$2\rho\alpha^2 B^2 L^2 \bar{\gamma}^2 + L\bar{\gamma} - 1 \leq 0. \tag{5.11}$$

In particular, the following choice of $\bar{\gamma}$ satisfies (5.11)*:*

$$\bar{\gamma} = \begin{cases} \frac{\sqrt{1+8\rho\alpha^2B^2}-1}{4\rho\alpha^2B^2L}, & B > 0, \\ 1/L, & B = 0. \end{cases} \tag{5.12}$$

Under Assumption 5.1, if $0 < \gamma \leq \bar{\gamma}$*,* ASAPP *converges to a first-order critical point with global sublinear rate. Specifically, after* K *total update steps,*

$$\min_{k\in\{0,\ldots,K-1\}} \mathbb{E}_{i_{0:K-1}}\left[\left\| \operatorname{grad} f(x^k) \right\|^2 \right] \leq \frac{2n(f(x^0)-f^\star)}{\gamma K}. \tag{5.13}$$

Remark 5.1. To the best of our knowledge, Theorem 5.1 establishes the first convergence result for asynchronous algorithms when solving a *non-convex* optimization problem over the product of matrix manifolds. While the existence of a convergent stepsize $\bar{\gamma}$ is of theoretical importance, we further note that its expression (5.12) offers the correct qualitative insights with respect to various problem-specific parameters, which we discuss next.

Relation with maximum delay (B): $\bar{\gamma}$ increases as maximum delay B decreases. Intuitively, as communication becomes increasingly available, each robot may take larger steps without causing divergence. The inverse relationship between $\bar{\gamma}$ and B is well known in the asynchronous optimization literature, and is first established by Bertsekas and Tsitsilis [58] in the Euclidean setting.

Relation with problem sparsity (ρ): $\bar{\gamma}$ increases as ρ decreases. Recall that $\rho \triangleq \Delta/n$ is defined as the ratio between the maximum number of neighbors a robot has and the total number of robots. Thus, ρ is a measure of *sparsity* of the robot-level graph $G_{\mathcal{R}}$. Intuitively, as $G_{\mathcal{R}}$ becomes more sparse, robots can use larger stepsize as their problems become increasingly decoupled. Such positive correlation between $\bar{\gamma}$ and problem sparsity has been a crucial feature in state-of-the-art asynchronous algorithms; see *e.g.*, [156].

Relation with problem smoothness (L): From (5.12), it can been seen that $\bar{\gamma}$ increases

asymptotically with $\mathcal{O}(1/L)$. Moreover, when there is no delay ($B = 0$), our stepsize matches the well-known constant of $1/L$ with which synchronous gradient descent converges to first-order critical points; see *e.g.*, [167].

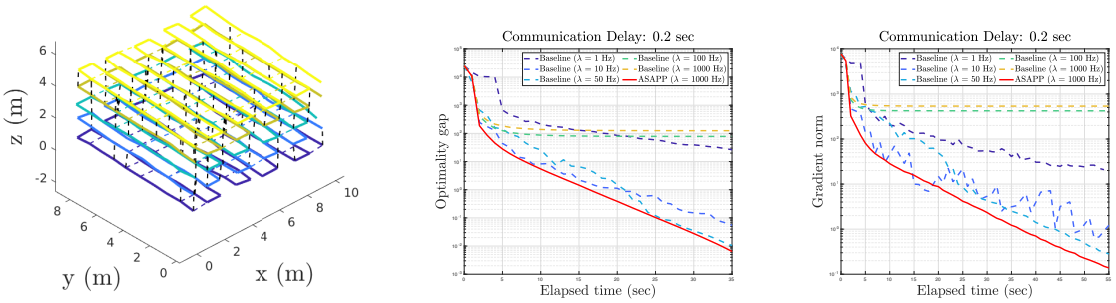
5.5 Experimental Results

We implement ASAPP in C++ and evaluate its performance on both simulated and real-world PGO datasets. We use ROPTLIB [184] for manifold related computations, and the Robot Operating System (ROS) [185] for inter-robot communication. The Poisson clock is implemented by halting the optimization thread after each iteration for a random amount of time exponentially distributed with rate λ (default to 1000 Hz). Since the time taken by each iteration is negligible, we expect the practical difference between this implementation and the theoretical model in Section 5.3.2 to be insignificant. All robots are simulated as separate ROS nodes running on a desktop computer with an Intel i7 quad-core CPU and 16 GB memory.

For each PGO problem, we use ASAPP to solve its rank-restricted relaxation (Problem 4.3) with $r = 5$. As is commonly done in prior work [155–161], in our experiments we select the stepsize empirically. During optimization, we record the evolution of the Riemannian gradient norm $\left\|\operatorname{grad} f(x^k)\right\|$, which measures convergence to a first-order critical point. In addition, we also record the optimality gap $f(x^k) - f(x^\star)$, where $x^\star$ is a global minimizer to PGO (Problem 2.2) computed by Cartan-Sync [20]. In Section 5.5.2, we also round the solution to $\mathrm{SE}(d)$ using the method in Section 4.6.2 and then compute the translation root mean squared error (RMSE) and rotation RMSE (in chordal distance) with respect to the global minimizer.

5.5.1 Evaluation in Simulation

We evaluate ASAPP in a simulated multi-robot SLAM scenario in which 5 robots move next to each other in a 3D grid with lawn mower trajectories (Figure 5-2a). Each robot has 100 poses. With probability 0.3, loop closures within and across trajectories are generated for poses within 1 m of each other. All measurements are



(a) Simulation (b) Optimality gap (c) Gradient norm

Figure 5-2: Performance evaluation on 5 robot simulation. The communication delay is fixed at 0.5 sec. We compare ASAPP (with stepsize $\gamma = 5 \times 10^{-4}$) with a baseline algorithm in which each robot uses Riemannian trust-region method to optimize its local variables. For a comprehensive evaluation, we run the baseline with varying optimization rate to record its performance under both synchronous and asynchronous regimes. (a) Example trajectories estimated by ASAPP, where trajectories of 5 robots are shown in different colors. Inter-robot measurements (loop closures) are shown as black dashed lines. (b) Optimality gap with respect to the centralized global minimizer $f(x^k) - f(x^\star)$. (c) Riemannian gradient norm $\|\text{grad}\, f(x^k)\|$.

corrupted by Langevin rotation noise with 2 deg standard deviation, and Gaussian translation noise with 0.05 m standard deviation. To minimize communication during initialization, we initialize the solution by propagating relative measurements along a spanning tree of the global pose graph. The stepsize used in simulation is $\gamma = 5{\times}10^{-4}$.

In the first experiment, we simulate communication delay by letting each robot communicate every 0.2 sec. We compare the performance of ASAPP (without preconditioning) against a baseline algorithm in which each robot uses the second-order Riemannian trust-region (RTR) method to optimize its local variable, similar to the approach in Chapter 4. Starting with SE-Sync [14], RTR has been used as the default solver in centralized or synchronous settings due to its global convergence guarantees and ability to exploit second-order geometry of the cost function. For a comprehensive evaluation, we record the performance of this baseline at different optimization rates (i.e. frequency at which robots update their local trajectories).

Figure 5-2b shows the optimality gaps achieved by the evaluated algorithms as a function of wall clock time. The corresponding reduction in the Riemannian gradient norm is shown in Figure 5-2c. ASAPP outperforms all variants of the baseline algorithm (dashed curves). We note that the behavior of the baseline algorithm is

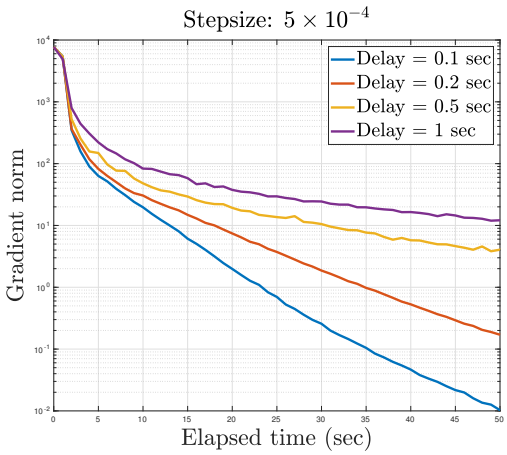


Figure 5-3: Convergence speed of ASAPP (stepsize $\gamma = 5 \times 10^{-4}$) under varying communication delay. As delay decreases, convergence becomes faster because robots have access to more up-to-date information from each other.

expected. At a low rate, e.g., $\lambda = 1$ Hz (dark blue dashed curve), the baseline algorithm is essentially synchronous as each robot has access to up-to-date poses from others. The empirical convergence speed is nevertheless slow, since each robot needs to wait for up-to-date information to arrive after each iteration. At a high rate, e.g., $\lambda = 1000$ Hz (dark yellow dashed curve), robots essentially behave asynchronously. However, since RTR does not regulate stepsize at each iteration, robots often significantly alter their solutions in the wrong direction (as a result of using outdated information), which leads to slow convergence or even non-convergence. In contrast, ASAPP is provably convergent, and furthermore is able to exploit asynchrony effectively to achieve speedup.

In addition, we also evaluate ASAPP under a wide range of communication delays. Figure 5-3 shows the speed of convergence in terms of gradient norm. We note that ASAPP converges in all cases, demonstrating its resilience against various delays in practice. Furthermore, as delay decreases, convergence becomes faster as robots have access to more up-to-date information from each other.

Dataset	# Poses	# Edges	Cost value f				Additional statistics for ASAPP			
			Opt. [20]	Initial	DGS [38]	ASAPP [ours]	Grad. Norm	Rot. RMSE [chordal]	Trans. RMSE [m]	Stepsize
CSAIL (2D)	1045	1171	31.47	36.10	31.54	**31.51**	0.36	0.0017	0.03	1.0
Intel (2D)	1228	1483	393.7	1205.9	394.6	**393.7**	0.002	2×10^{-6}	7×10^{-5}	1.0
Manhattan (2D)	3500	5453	193.9	1187.2	242.7	**227.7**	5.42	0.07	1.18	0.03
Garage (3D)	1661	6275	1.267	4.477	**1.277**	1.309	0.04	0.014	0.41	0.05
Sphere (3D)	2500	4949	1687.0	3075.7	1743.1	**1711.7**	2.73	0.02	0.50	0.23
Torus (3D)	5000	9048	24227	25812	24305	**24240**	8.38	0.017	0.07	1.0
Cubicle (3D)	5750	16869	717.1	916.2	**720.8**	734.5	12.67	0.030	0.08	0.065

Table 5.1: Evaluation on benchmark PGO datasets. Each dataset is divided into trajectories of 5 robots. We run ASAPP for 60 sec under a fixed communication delay of 0.1 sec. For reference, we also run DGS [38] for 600 synchronous iterations. We compare the final cost values of the two approaches, and highlight the better solution in bold. For ASAPP, we also report the used stepsize, achieved gradient norm, and rotation and translation root mean squared errors (RMSE) with respect to the global minimizer, computed by Cartan-Sync [20].

5.5.2 Evaluation on benchmark PGO datasets

We evaluate ASAPP on benchmark PGO datasets and compare its performance with Distributed Gauss-Seidel (DGS) [38], a state-of-the-art synchronous approach for distributed PGO used in recent multi-robot SLAM systems [39, 112]. Each dataset is divided into 5 segments simulating a collaborative SLAM mission with 5 robots.

Following Choudhary et al. [38], we initialize rotation estimates via distributed chordal initialization. To initialize the translations, we fix the rotation estimates and use distributed Gauss-Seidel to solve the reduced linear system over translations. To test on scenarios where accurate initializations are not available, we restrict the number of Gauss-Seidel iterations to 50 for both rotation and translation initialization.

Starting from the initial estimate, we run ASAPP with preconditioning for 60 sec, assuming for simplicity a fixed delay of 0.1 sec. Accordingly, we run DGS [38] on the problem (linearized at the initial estimate) for $60/0.1 = 600$ synchronous iterations. This setup favors DGS inherently, since each DGS iteration requires robots to communicate multiple times and update according to a specific order, which is likely to increase execution time in reality.

Table 5.1 compares the final cost values achieved by the two approaches, where for ASAPP we first round the solutions to $\mathrm{SE}(d)$. For ASAPP, we also report the used stepsize, final gradient norm, and estimation errors with respect to the

global minimizer computed by Cartan-Sync [20]. As our results show, ASAPP often compares favorably against DGS, especially when the quality of initialization is poor. This is an important advantage, as good initialization schemes (such as distributed chordal initialization developed in [38]) are usually iterative and thus expensive in terms of communication. Furthermore, the ASAPP solution is close to the global minimizer, except on the `Manhattan` dataset where the rotation and translation errors are relatively high.

We conclude this section by observing that on certain large datasets, convergence of ASAPP is slow as the iterate approaches a critical point. This is a consequence of the sublinear convergence rate, and in our case convergence is further impacted by the presence of communication delay. To address this, future work could consider accelerated methods (*e.g.*, the one presented in Chapter 4) to achieve higher precision.

5.6 Conclusion

This chapter presented ASAPP, an *asynchronous* and *provably delay-tolerant* algorithm to solve distributed pose graph optimization and its rank-restricted semidefinite relaxations. ASAPP enables each robot to run its local optimization process at a high rate, without waiting for updates from its peers over the network. Assuming a worst-case bound on the communication delay, we established the global first-order convergence of ASAPP, and showed the existence of a convergent stepsize whose value depends on the worst-case delay and inherent problem sparsity. When there is no delay, we further showed that this stepsize matches exactly with the corresponding constant in synchronous algorithms. Numerical evaluations on both simulation and real-world datasets confirm the advantages of ASAPP in reducing overall execution time, and demonstrate its resilience against a wide range of communication delay.

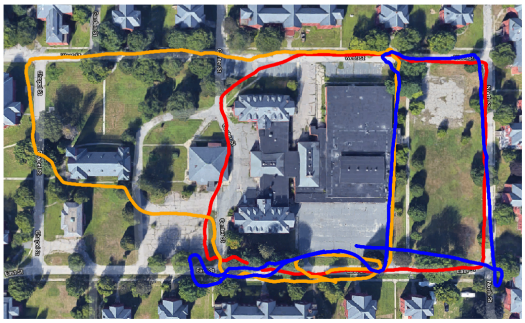
Chapter 6

Robust and Fully Distributed SLAM System and Large-Scale Field Experiments

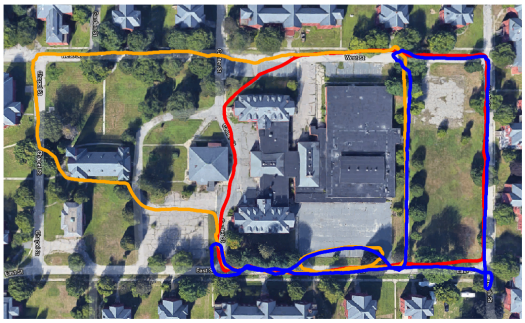
6.1 Introduction

The previous chapters focus on developing fully distributed optimization algorithms for collaborative PGO. In this chapter, we leverage these algorithms to develop a *complete* and *fully distributed system* for multi-robot collaborative SLAM (CSLAM), and demonstrate its usefulness on real-world datasets (*e.g.*, Figure 6-1) and large-scale field experiments (Section 6.7).

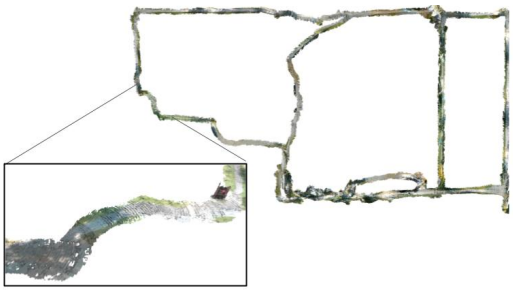
Contributions. The first contribution of this chapter is Kimera-Multi, **a fully distributed system for multi-robot dense metric-semantic SLAM**. Our system enables a team of robots to collaboratively estimate a semantically annotated 3D mesh model of the environment in real time. Each robot runs Kimera [36] to process onboard visual-inertial sensor data and obtain local trajectory and 3D mesh estimates. When communication becomes available, a fully distributed procedure is triggered to perform inter-robot place recognition, relative pose estimation, and robust distributed trajectory estimation. From the jointly optimized trajectory estimates, each robot performs real-time local mesh deformation to correct local mapping drift and improve



(a) Kimera-VIO trajectory estimate



(b) Kimera-Multi trajectory estimate



(c) Kimera-Multi optimized mesh

Figure 6-1: Demonstration of Kimera-Multi in a three-robot collaborative SLAM dataset collected at Medfield, Massachusetts, USA. Total trajectory length (including all robots) is 2188 meters. (a) Trajectory estimate from Kimera-VIO is affected by estimation drift. (b) Kimera-Multi achieves accurate and robust trajectory estimation. (c) Kimera-Multi also produces an optimized 3D mesh of the environment.

global map consistency. The implementation of Kimera-Multi is modular and allows different components to be disabled or replaced.

The second technical contribution of this chapter is a **new, two-stage method for outlier-robust distributed pose graph optimization (PGO)**, which serves as the distributed back-end of Kimera-Multi. The first stage initializes robots' local trajectories in a global reference frame by using graduated non-convexity (GNC) [28]

to estimate relative transformations between the coordinate frames of pairs of robots. This method is robust to outlier loop closures and, furthermore, is efficient because it does not require iterative communication. The second stage solves the full robust PGO problem. For this purpose, we present a distributed extension of GNC built on top of the RBCD solver developed in Chapter 4. Compared to prior techniques, our approach achieves more robust and accurate trajectory estimation, and is less sensitive to parameter tuning.

Lastly but not least, this chapter presents **extensive quantitative evaluation and large-scale field experiments** of Kimera-Multi. First, we validate the accuracy, robustness, and communication efficiency of Kimera-Multi using a collection of photo-realistic simulations and real-world datasets. Furthermore, we describe large-scale live experiments conducted on the MIT campus along with the challenging large-scale benchmarking datasets compiled from data recorded during the live experiments. We provide quantitative results from controlled experiments and discuss lessons learned from live field tests. We release the source code of Kimera-Multi[1] and all datasets[2] together with accurate reference trajectories and point cloud maps to facilitate further research in this area.

The remainder of this chapter is organized as follows. Section 6.2 presents an overview of the proposed system. Sections 6.3 to 6.5 describe the individual system components, including the distributed front-end for inter-robot loop closure detection, the distributed back-end for robust trajectory estimation, and the local mesh optimization for locally dense metric-semantic mapping. Section 6.6 presents offline evaluation on photo-realistic simulations and real-world datasets. Lastly, Section 6.7 presents field experiments where Kimera-Multi is deployed on 6-8 ground robots that perform CSLAM in large-scale mixed indoor-outdoor environments.

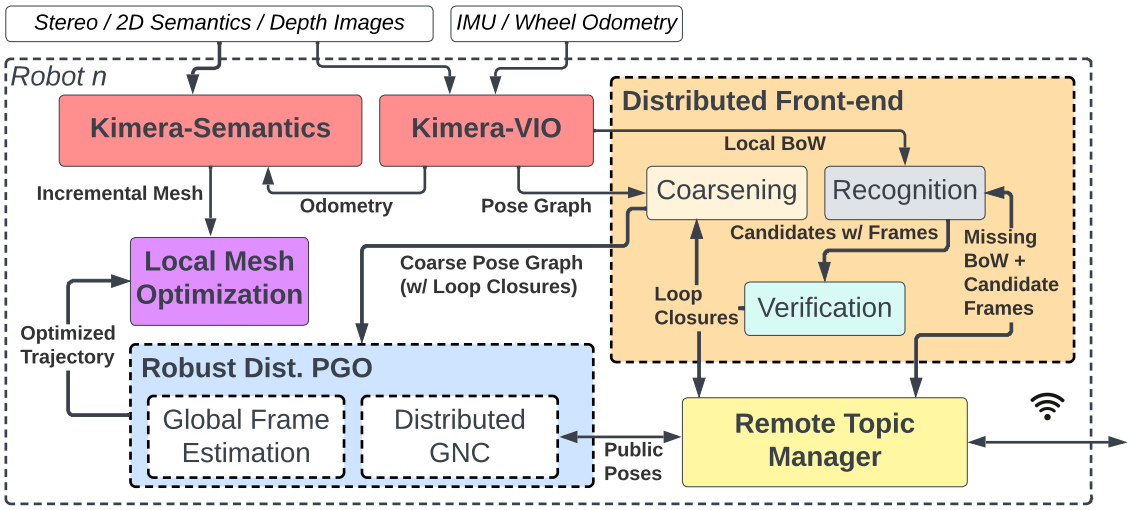


Figure 6-2: Kimera-Multi: system architecture. Each robot runs Kimera (including Kimera-VIO and Kimera-Semantics) to estimate local trajectory and mesh. Robots then communicate to perform distributed loop closure detection and robust distributed PGO. Given the optimized trajectory, each robot performs local mesh optimization. All inter-robot communication is handled by the remote topic manager module.

6.2 System Overview

In Kimera-Multi, each robot runs the *fully decentralized* metric-semantic SLAM system shown in Fig. 6-2. The system consists of the following main modules: (i) local (single-robot) Kimera, (ii) distributed loop closure detection, (iii) robust distributed trajectory estimation via PGO, (iv) local mesh optimization, and (v) remote topic manager. Among these modules, distributed loop closure detection and robust distributed PGO are the only ones that involve communication between robots, and the detailed communication is managed by the remote topic manager. Fig. 6-2 shows the data flow between these modules.

Kimera [36] runs onboard each robot and provides real-time local trajectory and mesh estimation. In particular, Kimera-VIO [186] serves as the visual-inertial odometry module, which processes raw stereo images and IMU data to obtain an estimate of the odometric trajectory of the robot. Kimera-Semantics [186] processes depth images (possibly obtained from RGB-D cameras or by stereo matching) and

[1] https://github.com/MIT-SPARK/Kimera-Multi
[2] https://github.com/MIT-SPARK/Kimera-Multi-Data

2D semantic segmentations [187] and produces a dense metric-semantic 3D mesh using the VIO pose estimates. In addition, Kimera-VIO computes a Bag-of-Words (BoW) representation of each keyframe using ORB features and DBoW2 [46], which is used for distributed loop closure detection. Interested readers are referred to [36, 186] for more technical details.

Distributed Loop Closure Detection (Section 6.3) is executed whenever two robots α and β are within communication range. The robots exchange BoW descriptors of the keyframes they collected. When the robots find a pair of matching descriptors (typically corresponding to observations of the same place), they perform relative pose estimation using standard geometric verification techniques. The relative pose corresponds to a putative inter-robot loop closure, and is used during robust distributed trajectory estimation.

Robust Distributed Trajectory Estimation (Section 6.4) solves for the optimal trajectory estimates of all robots in a global reference frame, by performing robust distributed PGO using odometric measurements from Kimera-VIO and all putative loop closures detected so far. At the beginning, a robust initialization scheme is used to find coarse relative transformations between robots' reference frames. Then, a robust optimization procedure based on a distributed extension of GNC [28] using the RBCD solver developed in Chapter 4 is employed to simultaneously select inlier loop closures and recover optimal trajectory estimates. Compared to the incremental PCM technique [81] used in an earlier version of this system [23], our new approach enables more robust and accurate trajectory estimation, and is less sensitive to parameter tuning.

Local Mesh Optimization (LMO; Section 6.5) is executed after the robust distributed trajectory estimation stage. This module performs a local processing step that deforms the mesh at each robot to enforce consistency with the trajectory estimate resulting from distributed PGO.

Inter-Robot Communication. Kimera-Multi implements a *remote topic manager* module to handle communication between robots. This module closely integrates with the Robot Operating System (ROS) [185] publish-subscribe paradigm and man-

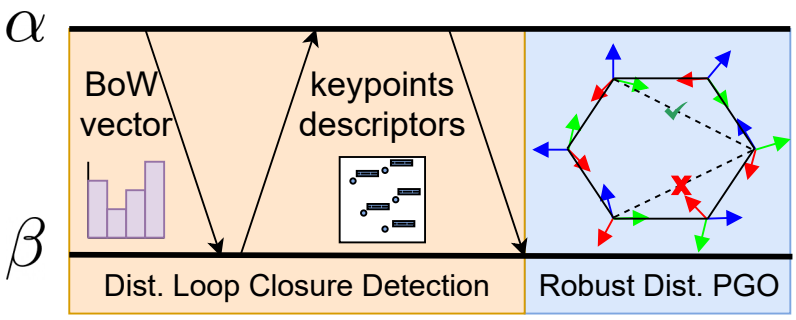


Figure 6-3: Communication protocol and data flow between pair of robots.

ages incoming and outgoing ROS messages with other robots. The remote topic manager also keeps track of currently connected robots, and initiates new connections when others are within communication range. The remote topic manager is implemented using the open-source ENet library,[3] which provides lightweight communication using UDP and supports reliable, in-order transmission of selected data streams. Lastly, ENet also provides diagnostic statistics such as delay and packet loss that are helpful for evaluating communication performance.

The entire Kimera-Multi implemented in C++.[4] The system runs online using a CPU and is modular, thus allowing modules to be replaced or removed. For instance, the system can also produce a dense *metric* mesh if semantic labels are not available, or only produce the optimized trajectory if the dense reconstruction is not required by the user.

6.3 Distributed Loop Closure Detection

This section describes the front-end of Kimera-Multi, which is responsible for detecting inter-robot loop closures between pairs of robots. The information flow is summarized in Fig. 6-3. Whenever two robots can communicate, one of the robots executes the Kimera-Multi front-end to detect inter-robot loop closures.[5] The front-end consists of three main components: *place recognition*, *geometric verification*, and *pose graph*

[3] http://enet.bespin.org/

[4] Source code is available at https://github.com/MIT-SPARK/Kimera-Multi.

[5] In our implementation, between each pair of robots, the robot with a smaller ID is designated to run the front-end.

coarsening.

The *place recognition* component subscribes to BoW vectors from the other robot, and finds matches by searching the local database of past BoW vectors. To avoid a high bandwidth usage, we perform an optional downsampling and only transmit every n_bth BoW vector (default $n_b = 3$). A candidate loop closure is identified if the normalized similarity score is higher than a threshold α (default 0.5). In order to operate robustly under sporadic communication, our implementation tolerates *out-of-order* arrival of BoW vectors. Each robot also periodically examines its local database for missing BoW vectors, and publishes a request to the connected robot that has the most missing BoW vectors.

The *geometric verification* component processes the queue of candidate loop closures. For each candidate inter-robot loop closure, a request is transmitted to the other robot to send its corresponding visual keyframe, which contains 3D keypoints and ORB descriptors. This robot then performs standard descriptor matching and computes the underlying relative transformation using monocular and stereo RANSAC. The estimated transformation is sent back to the other robot to be included in its local pose graph.

To prevent the rapid growth of the multi-robot pose graph, we add a *pose graph coarsening* component that subscribes to the local pose graph and loop closures, and then reduces the graph by aggregating pose variables within a specified distance d (default $d = 2$m). In our experiments, the coarsened pose graph is on average 90% smaller than the original pose graph, which enables more efficient optimization in our back-end.

6.4 Robust Distributed Trajectory Estimation

In Kimera-Multi, the robots estimate their trajectories by collaboratively solving a PGO problem using the entire team's odometry measurements and intra-robot and inter-robot loop closures. Some of these loop closures may be outliers (due to, *e.g.*, perceptual aliasing) and thus we need an outlier-robust method for solving PGO.

Many prior systems (including an earlier version of Kimera-Multi [23]) use PCM [81] for outlier rejection via maximum clique computation prior to trajectory estimation. However, even with parallelization [188], the runtime of exact maximum clique search exceeds 10 seconds already in graphs with 700 loop closures, which is not practical for our application. For this reason, in practice PCM has to rely on heuristic maximum clique algorithms and thus often exhibits poor recall, as shown in Section 6.6.1.

We propose a new distributed approach for robust trajectory estimation based on GNC [28]. The main idea in GNC is to start from a convex approximation of the robust cost function and then gradually introduce the non-convexity to prevent convergence to spurious solutions. While in general GNC does not require an initial guess [28], it has been observed that global solvers for 3D SLAM (*e.g.*, SE-Sync [14]) become too slow in the presence of outliers [116]. For this reason, in [116] local optimization is performed instead at each iteration of GNC (starting from an outlier-free initial guess), and this approach has been shown to be very effective. In single-robot SLAM, one can easily obtain an outlier-free initial guess by chaining together odometry measurements. In the multi-robot case, there is no odometry between different robots' poses, and the challenge thus becomes building an initial guess that is insensitive to outliers.

To address the aforementioned challenge, the proposed distributed graduated non-convexity (D-GNC) approach involves two stages. In the first stage (Section 6.4.2), we use an outlier-robust and communication-efficient method to initialize robots' trajectories in a global reference frame. In the second stage (Section 6.4.3), we develop a fully distributed procedure to execute GNC, using the RBCD distributed solver as a subroutine. We discuss several additional implementation details in Section 6.4.4. Algorithm 6.1 provides the pseudocode of D-GNC.

6.4.1 Background: Graduated Non-Convexity

We start by providing a brief review of GNC [28, 189]. One challenge associated with classical M-estimation [190, 191] is that the employed robust cost function ρ can be highly non-convex, hence making local search techniques sensitive to the initial guess.

Algorithm 6.1 Distributed Graduated Non-Convexity (D-GNC)

Input:

- Initial trajectory estimates in *local* frames of each robot
- Odometry and intra-robot and inter-robot loop closures that each robot is involved in
- Threshold $\bar{c}$ of truncated least squares (TLS) cost

Output:

- Optimized trajectory estimate of each robot in global frame

1: **Robust initialization**: robots communicate to initialize trajectory estimates in a global reference frame (Section 6.4.2).
2: In parallel, each robot initializes GNC weights for its *local* intra and inter-robot loop closures $w_i = 1, \forall i$.
3: **while** not converged **do**
4: **Variable update**: with fixed weights, robots communicate to execute RBCD for T iterations (default $T = 15$).
5: **Weight update**: in parallel, each robot updates GNC weights for intra-robot loop closures and inter-robot loop closures it is involved in.
6: **Parameter update**: in parallel, each robot updates the control parameter μ.
7: **end while**

The key idea behind GNC is to optimize a sequence of easier (*i.e.*, less non-convex) surrogate cost functions that gradually converges to the original robust cost function. Each surrogate problem takes the same form as classical M-estimation,

$$\min_{x \in \mathcal{X}} \sum_i \rho_\mu(r_i(x)), \tag{6.1}$$

where $r_i : \mathcal{X} \to \mathbb{R}$ is the residual error associated with the ith measurement. The sequence of surrogate functions ρ_μ, parameterized by control parameter μ, satisfies that for some given constants μ_0 and μ_1: (i) for $\mu \to \mu_0$, the function ρ_μ is convex, and (ii) for $\mu \to \mu_1$, ρ_μ converges to the original (non-convex) robust cost function ρ. In practice, one initializes μ near μ_0, and gradually updates its value to approach μ_1 as optimization proceeds.

For each instance of (6.1), GNC reformulates the problem using the Black-Rangarajan Duality [189], which states that under certain technical conditions (satisfied by all common choices of robust cost functions), (6.1) is equivalent to the following opti-

mization problem,

$$\min_{x\in\mathcal{X},w_i\in[0,1]} \sum_i \left[w_i r_i^2(x) + \Phi_{\rho_\mu}(w_i)\right], \tag{6.2}$$

where $w_i \in [0,1]$ is a scalar weight associated with the ith measurement. In (6.2), the *outlier process* $\Phi_{\rho_\mu}(w_i)$ introduces a penalty term for each w_i, and its expression depends on the chosen robust cost function ρ and the control parameter μ. Similar to the classical iterative reweighted least squares (IRLS) scheme, GNC performs alternating minimization over the variable x and weights w_i to optimize (6.2), but in the meantime also updates the control parameter μ:

1. **Variable update:** Minimize (6.2) with respect to x with fixed weights w_i. This amounts to solving a standard weighted least-squares problem,

$$x^\star \in \arg\min_{x\in\mathcal{X}} \sum_i w_i r_i^2(x). \tag{6.3}$$

2. **Weight update:** Minimize (6.2) with respect to w_i with fixed variable x. The corresponding update for each w_i has a closed-form expression that depends on the current robust surrogate function ρ_μ; see [28, Proposition 3-4].

3. **Parameter update:** Update μ by a constant factor to approach μ_1.

The control parameter μ is initialized at a value close to μ_0. In the absence of a better guess, all weights are initialized to one (*i.e.*, all measurements are considered inliers initially). Then the steps above are repeated until μ approaches μ_1.

6.4.2 Robust Distributed Initialization

To optimize the pose graph, we first need to initialize all robot poses in a shared (global) coordinate frame (Algorithm 6.1, line 1). Each robot can readily initialize its trajectory in its *local* reference frame by chaining odometry measurements. To express these local initial guesses in the global reference frame, however, we must estimate the relative pose between the local reference frames.

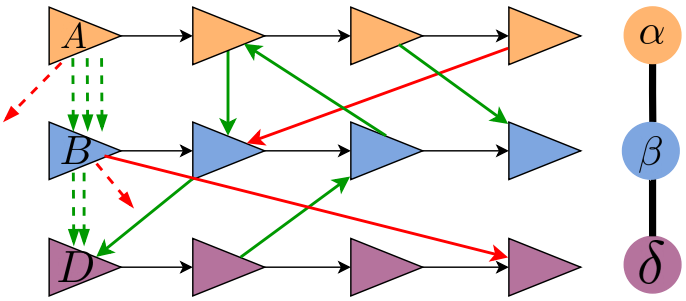


Figure 6-4: Robust distributed initialization. **Left**: Three-robot scenario with local reference frames A, B, D, each coinciding with the first pose of the corresponding robot. Between every pair of robots, inlier loop closures ($\longrightarrow$) lead to similar estimates for the alignment between frames ($\dashrightarrow$). Each outlier loop closure ($\longrightarrow$) produces an outlier frame alignment ($\dashrightarrow$), which can be rejected with GNC. **Right**: Corresponding robot-level spanning tree.

Pairwise coordinate frame estimation. First, let us see how this can be done between two robots α and β, with local reference frames A and B, respectively. Consider a loop closure between the ith pose of α and jth pose of β, denoted as $\widetilde{\boldsymbol{X}}_{\beta_j}^{\alpha_i} \in \mathrm{SE}(3)$. Denote the odometric estimates of pose i and j (in the *local* frames of the two robots) as $\widehat{\boldsymbol{X}}_{\alpha_i}^{A}, \widehat{\boldsymbol{X}}_{\beta_j}^{B} \in \mathrm{SE}(3)$. By combining these pose estimates with the loop closure, we obtain a noisy estimate of the relative transformation between frames A and B,

$$\widehat{\boldsymbol{X}}_{B_{ij}}^{A} \triangleq \widehat{\boldsymbol{X}}_{\alpha_i}^{A} \widetilde{\boldsymbol{X}}_{\beta_j}^{\alpha_i} \big(\widehat{\boldsymbol{X}}_{\beta_j}^{B}\big)^{-1}, \tag{6.4}$$

where the subscript of $\widehat{\boldsymbol{X}}_{B_{ij}}^{A}$ indicates that this estimate is computed using loop closure (i, j). From (6.4), we see that each inter-robot loop closure provides a candidate alignment for the reference frames A and B. Furthermore, candidate alignments produced by inlier loop closures are expected to be in mutual agreement; see Fig. 6-4 and also [117]. To obtain a reliable estimate of the true relative transformation, we thus formulate and solve the following *robust* pose averaging problem,

$$\widehat{\boldsymbol{X}}_{B}^{A} \in \underset{\boldsymbol{X} \in \mathrm{SE}(3)}{\arg\min} \sum_{(i,j) \in L_{\alpha,\beta}} \rho(r_{ij}(\boldsymbol{X})), \tag{6.5}$$

where $\rho : \mathbb{R} \to \mathbb{R}$ is the truncated least squares (TLS) robust cost function [28], and $L_{\alpha,\beta}$ is the set of inter-robot loop closures between robot α and β. Each residual measures the geodesic distance between the to-be-computed average pose $\boldsymbol{X}$ and the

measurement $\widehat{\boldsymbol{X}}_{B_{ij}}^{A}$,

$$r_{ij}(\boldsymbol{X}) \triangleq \left\| \boldsymbol{X} \boxminus \widehat{\boldsymbol{X}}_{B_{ij}}^{A} \right\|_{\Sigma}, \tag{6.6}$$

where $\Sigma \in \mathbb{S}_{++}^{6}$ is a fixed covariance matrix. In our implementation, we use a diagonal covariance with a standard deviation of 0.1 rad for rotation and 0.5 m for translation. Between a given pair of robots, one robot can solve (6.5) *locally* using GNC [28] without extra communication (since each robot already has access to all loop closures it is involved in), and transmits the solution to the other robot. In practice, we use the GNC implementation available in GTSAM [6], which uses Levenberg-Marquardt (initialized at identity pose) in each GNC variable update to solve (6.5). The estimated transformation is accepted if the solution is supported by at least 3 inlier loop closures.

Multi-robot coordinate frame estimation. The above *pairwise* procedure can be executed repeatedly to express *all* local reference frames (and trajectory estimates) in a global frame while being robust to outliers. To do so, we first choose an arbitrary spanning tree in the robot-level dependency graph first introduced in Section 2.2.2. Recall that this is a graph whose vertices correspond to robots and edges represent the presence of at least one inter-robot loop closure between the two corresponding robots (Fig. 6-4). Note that the spanning tree induces a unique path between any two robots. We select an arbitrary robot α and use its reference frame A as the global frame.[6] For each remaining robot β, we need to obtain its relative transformation to the global frame $\widehat{\boldsymbol{X}}_{B}^{A} \in \mathrm{SE}(3)$. This is done by traversing the unique path in the robot-level spanning tree from α to β, and composing all estimated *pairwise* transformations computed using (6.5) along the way. In practice, this procedure can be performed in a fully distributed fashion, by incrementally growing the robot-level spanning tree from α using local communication. Finally, each robot β uses its corresponding $\widehat{\boldsymbol{X}}_{B}^{A}$ to express its initial trajectory in the global frame. Note that our distributed PGO approach does not require the robots to share these initial trajectory estimates, but

[6] In our implementation, the reference frame of the robot with the smallest ID is used as the global frame.

only requires them to be expressed in a shared global frame at each robot.

6.4.3 Robust Distributed Pose Graph Optimization

Following the initialization stage, robots perform robust distributed PGO to obtain optimal trajectory estimates while simultaneously rejecting outlier loop closures. Let $\boldsymbol{X}_{\alpha_i} = (\boldsymbol{R}_{\alpha_i}, \boldsymbol{t}_{\alpha_i}) \in \mathrm{SE}(3)$ denote the ith pose of robot α in the global frame. We aim to optimize all pose variables using all odometric measurements and putative loop closures.

$$\min_{\substack{\boldsymbol{X}_{\alpha_i}\in\mathrm{SE}(3),\\ \forall\alpha\in\mathcal{R},\,\forall i}} \underbrace{\sum_{\alpha\in\mathcal{R}}\sum_{i=1}^{n_\alpha-1} r_{\alpha_i}(\boldsymbol{X}_{\alpha_i}, \boldsymbol{X}_{\alpha_{i+1}})^2}_{\text{odometry}} + \underbrace{\sum_{(\alpha_i,\beta_j)\in L} \rho\big(r_{\beta_i}^{\alpha_i}(\boldsymbol{X}_{\alpha_i}, \boldsymbol{X}_{\beta_j})\big)}_{\text{loop closures}}, \tag{6.7}$$

where $\mathcal{R} = \{\alpha, \beta, \ldots\}$ denotes the set of robots, n_α is the total number of poses of robot α, and the set of loop closures L includes both intra-robot and inter-robot loop closures. Each residual error in (6.7) corresponds to a single relative pose measurement in the global pose graph, where the residual error is measured using the chordal distance. For example, the residual corresponding to a loop closure is given by [14],

$$r_{\beta_i}^{\alpha_i}(\boldsymbol{X}_{\alpha_i}, \boldsymbol{X}_{\beta_j}) \triangleq \left(w_R \left\| \boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i}\widetilde{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_F^2 + w_t \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} - \boldsymbol{R}_{\alpha_i}\widetilde{\boldsymbol{t}}_{\beta_j}^{\alpha_i} \right\|_2^2 \right)^{1/2}, \tag{6.8}$$

where $\widetilde{\boldsymbol{X}}_{\beta_j}^{\alpha_i} = (\widetilde{\boldsymbol{R}}_{\beta_j}^{\alpha_i}, \widetilde{\boldsymbol{t}}_{\beta_j}^{\alpha_i}) \in \mathrm{SE}(3)$ is the observed noisy transformation, and $w_R, w_t > 0$ specify measurement precisions. We employ the standard quadratic cost for odometric measurements as they are outlier-free. For loop closures, we choose ρ to be the TLS function as in Section 6.4.2.

To solve (6.7), we develop a *fully distributed* variant of GNC which uses the RBCD solver developed in Chapter 4 as the workhorse during iterative optimization. Recall from Section 6.4.1 that GNC alternates between variable (*i.e.*, trajectory) updates and weight updates. In the following, we discuss how each of these two operations are performed in the distributed setup.

Variable update. In this case, the variable update step becomes an instance of

standard (weighted) PGO,

$$\min_{\substack{\boldsymbol{X}_{\alpha_i}\in \mathrm{SE}(3),\\ \forall \alpha\in\mathcal{R},\,\forall i}} \quad \sum_{\alpha\in\mathcal{R}}\sum_{i=1}^{n_\alpha-1} r_{\alpha_i}(\boldsymbol{X}_{\alpha_i},\boldsymbol{X}_{\alpha_{i+1}})^2 + \sum_{(\alpha_i,\beta_j)\in L} w_{\beta_j}^{\alpha_i}\cdot r_{\beta_i}^{\alpha_i}(\boldsymbol{X}_{\alpha_i},\boldsymbol{X}_{\beta_j})^2. \tag{6.9}$$

Compared to (6.7), terms including the robust cost function ρ (corresponding to the loop closures) are replaced by weighted squared residuals; see also (6.3). We apply the RBCD solver for distributed optimization of (6.9) (Algorithm 6.1, line 4). In the following, we recall several properties of RBCD that are discussed in depth in Chapter 4. RBCD operates on the rank-restricted relaxation [14] of (6.9) and subsequently projects the solution to the Special Euclidean group. In our implementation, we set the default rank relaxation to 5. In RBCD, each robot $\alpha \in \mathcal{R}$ is responsible for estimating its own trajectory $\boldsymbol{X}_\alpha \triangleq \{\boldsymbol{X}_{\alpha_i}, i = 1,\dots,n_\alpha\}$. During execution, robots alternate to update their trajectories by relying on partial information exchange with their teammates. Specifically, at each iteration in which robot α updates its trajectory, it needs to communicate once with its neighboring robots (*i.e.*, robots that share inter-robot loop closures with robot α), where the communication can be either direct or relayed by other robots. Furthermore, robot α only needs to receive neighboring robots' "public poses" (*i.e.*, poses that share inter-robot loop closures with robot α). This property allows RBCD to preserve privacy and saves communication effort over the remaining poses.

In the original (centralized) GNC algorithm, each variable update step is solved to full convergence using a global solver or local search technique. In the distributed setup, however, solving each instance of (6.9) to full convergence can be slow, due to the first-order nature of typical distributed optimization methods (including both RBCD and DGS [38]). To develop a more practical and efficient approach, we relax the convergence requirements and allow *approximate* solutions during variable updates. Specifically, our implementation supports two strategies for performing approximate optimization. The first strategy (used in Section 6.6) is to run RBCD for a fixed number of iterations (default 15) and use the resulting trajectory estimates to warm start the next variable update step. The second strategy (used in Section 6.7)

is to run RBCD until reaching a (potentially loose) convergence threshold. In our implementation, we terminate optimization once the relative changes of all robots' translation estimates are less than a threshold ϵ_{rel} (default 0.2m).

Weight update. In the original GNC paper [28], it has been shown that the weight update for each residual function using TLS only depends on the current residual error $\widehat{r}_i$, control parameter μ, and the threshold $\bar{c}$ of the TLS cost,

$$w_i \leftarrow \begin{cases} 0, & \text{if } \widehat{r}_i^2 \in \left[\frac{\mu+1}{\mu}\bar{c}^2, +\infty\right], \\ \frac{\bar{c}}{\widehat{r}_i}\sqrt{\mu(\mu+1)} - \mu, & \text{if } \widehat{r}_i^2 \in \left[\frac{\mu}{\mu+1}\bar{c}^2, \frac{\mu+1}{\mu}\bar{c}^2\right], \\ 1, & \text{if } \widehat{r}_i^2 \in \left[0, \frac{\mu}{\mu+1}\bar{c}^2\right]. \end{cases} \tag{6.10}$$

See [28, Proposition 4] for more details. The weight update step is particularly suitable for distributed computation, as (6.10) suggests that this operation can be performed *independently and in parallel* for each residual function (*i.e.*, loop closure). We leverage this insight to implement a fully distributed weight update scheme (Algorithm 6.1, line 5). Specifically, each robot first updates weights associated with its internal loop closures in parallel. Then, for each inter-robot loop closure, one of the two involved robots computes the updated weight, and subsequently transmits the new weight to the other robot. After the weight update stage, each robot also updates its local copy of the control parameter μ, so that the sequence of surrogate cost functions gradually converges to the original TLS function (Algorithm 6.1, line 6).

6.4.4 Implementation Details

Standard formulation of distributed PGO assumes that all robots are available and connected, so that the team can collaborate to solve the entire multi-robot PGO problem. However, this may be a strong assumption in practice, as robots may form multiple disconnected clusters at any time (*e.g.*, due to sporadic communication). To address this challenge, our implementation allows concurrent PGO within multiple clusters of connected robots. To maintain consistency across clusters, a prior over

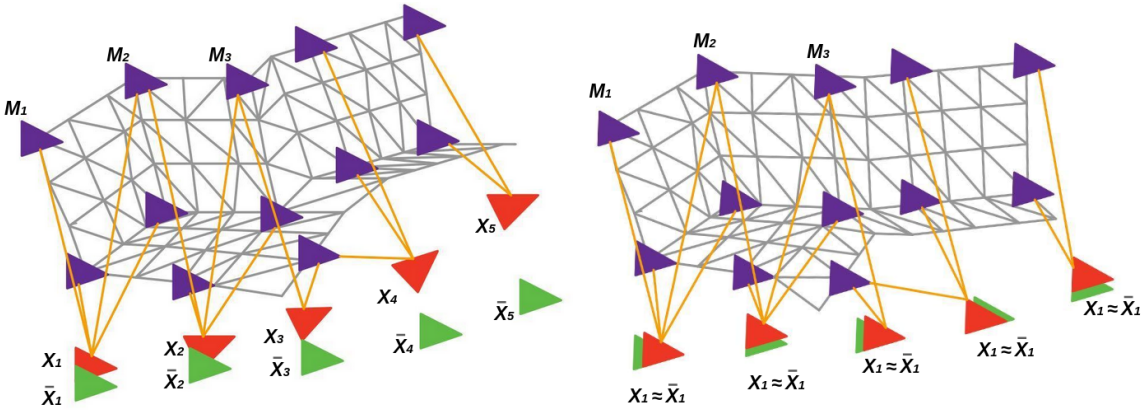
the global reference frame is added to each cluster.[7] Within each cluster, robots iteratively refine their trajectory estimates by performing local optimization steps and communicating public poses as described above. For this purpose, we assume that robots within the same cluster can reach each other (either via direct links or additional routing), so that they can transmit their public poses to the intended destination and maintain synchronization during distributed optimization.

6.5 Local Mesh Optimization

This section describes how to perform local correction of the 3D mesh in response to a loop closure. Kimera-Semantics builds the 3D mesh from the Kimera-VIO (odometric) estimate. However, since distributed PGO described in previous section improves the accuracy of the trajectory estimate by enforcing loop closures, it is desirable to correct the mesh according to the optimized trajectory estimate (*i.e.*, each time distributed PGO is executed). Here we propose an approach for mesh optimization based on deformation graphs [192]. *Deformation graphs* are a model from computer graphics that deforms a given mesh in order to anchor points in this mesh to user-defined locations while ensuring that the mesh remains locally rigid; deformation graphs are typically used for 3D animations, where one wants to animate a 3D object while ensuring it moves smoothly and without artifacts [192].

Creating the deformation graph. In our approach, we create a unified deformation graph including a simplified mesh and a pose graph of trajectory keyframes. The process is illustrated in Fig. 6-5. The intuition is that the "anchor points" in [192] will be the keyframes in our trajectory. More specifically, while Kimera-Semantics builds a local 3D mesh for each robot α using pose estimates from Kimera-VIO, we keep track of the subset of 3D mesh vertices seen in each keyframe from Kimera-VIO. To build the deformation graph, we first subsample the mesh from Kimera-Semantics to obtain a simplified mesh. We simplify the mesh with an online vertex clustering method by

[7]The prior is obtained from the last round of distributed optimization involving the first robot (who we assign as the global reference frame). Within each cluster, the prior is added to the robot with the smallest ID.



(a) Undeformed mesh.

(b) Deformed mesh

Figure 6-5: LMO deformation graph including mesh vertices (violet) and keyframe vertices (red). Edges connect two mesh vertices that are adjacent in the mesh (gray links), as well as mesh vertices with the keyframe vertices they are observed in (orange links). The green poses denote optimized poses from distributed PGO.

storing the vertices of the mesh in an octree data structure; as the mesh grows, the vertices in the same voxel of the octree are merged and degenerate faces and edges are removed. The voxel size is tuned according to the environment or the dataset. Then, the vertices of this simplified mesh and the corresponding keyframe poses are added as *vertices* in the deformation graph; we are going to refer to the corresponding vertices in the deformation graph as *mesh vertices* and *keyframe vertices*. Moreover, we add two types of *edges* to the deformation graph: *mesh edges* (corresponding to pairs of mesh vertices sharing a face in the simplified mesh), and *keyframe edges* (connecting a keyframe with the set of mesh vertices it observes).

For each mesh vertex k in the deformation graph, we assign a transformation $\boldsymbol{M}_k = (\boldsymbol{R}_k^M, \boldsymbol{t}_k^M)$, where $\boldsymbol{R}_k^M \in \mathrm{SO}(3)$ and $\boldsymbol{t}_k^M \in \mathbb{R}3$; $\boldsymbol{M}_k$ defines a local coordinate frame, where $\boldsymbol{R}_k$ is initialized to the identity and $\boldsymbol{t}_k$ is initialized to the position $\boldsymbol{g}_k$ of the mesh vertex from Kimera-Semantics (*i.e.*, without accounting for loop closures). We also assign a pose $\boldsymbol{X}_i = (\boldsymbol{R}_k^x, \boldsymbol{t}_k^x)$ to each keyframe vertex i. The pose is initialized to the pose estimates from Kimera-VIO.

Optimizing the deformation graph. The goal is to correct the mesh on each robot in response to changes in the keyframe poses (due to PGO). Towards this goal, we need to adjust the poses (and the mesh vertex positions) to "anchor" the keyframe

poses to the latest estimates from distributed PGO as shown in Fig. 6-5. Denoting the optimized poses from distributed PGO as $\bar{\boldsymbol{X}}_i$, and calling n the number of keyframes in the trajectory and m the total number of mesh vertices in the deformation graph. Following [192], we compute updated poses $\boldsymbol{X}_i$, $\boldsymbol{M}_k$ of the vertices in the deformation graph by solving the following *local* optimization problem at each robot,

$$\begin{aligned}\underset{\substack{\boldsymbol{X}_1,\dots,\boldsymbol{X}_n\in\mathrm{SE}(3)\\ \boldsymbol{M}_1,\dots,\boldsymbol{M}_m\in\mathrm{SE}(3)}}{\arg\min} & \sum_{i=0}^{n} ||\boldsymbol{X}_i \boxminus \bar{\boldsymbol{X}}_i||^2_{\Sigma_x} + \\ & \sum_{k=0}^{m}\sum_{l\in\mathcal{N}^M(k)} ||\boldsymbol{R}^M_k(\boldsymbol{g}_l-\boldsymbol{g}_k)+\boldsymbol{t}^M_k-\boldsymbol{t}^M_l||^2_{\Sigma}+ \\ & \sum_{i=0}^{n}\sum_{l\in\mathcal{N}^M(i)} ||\boldsymbol{R}^x_i\widetilde{\boldsymbol{g}}_{il}+\boldsymbol{t}^x_i-\boldsymbol{t}^M_l||^2_{\Sigma} \end{aligned} \tag{6.11}$$

where $\boldsymbol{g}_k$ denotes the non-deformed position of vertex k in the deformation graph, $\widetilde{\boldsymbol{g}}_{il}$ denotes the non-deformed position of vertex l in the coordinate frame of keyframe i, $\mathcal{N}^M(k)$ denotes all the mesh vertices in the deformation graph connected to vertex k, and $\boxminus$ denotes a tangent-space representation of the relative pose between $\boldsymbol{X}_i$ and $\bar{\boldsymbol{X}}_i$ [, 7.1]. Intuitively, the first term in the minimization (6.11) enforces ("anchors") the poses of each keyframe $\boldsymbol{X}_i$ to match the optimized poses $\bar{\boldsymbol{X}}_i$ from distributed PGO. The second term enforces local rigidity of the mesh by minimizing the mismatch with respect to the non-deformed configuration $\boldsymbol{g}_k$. The third term enforces local rigidity of the relative positions between keyframes and mesh vertices by minimizing the mismatch with respect to the non-deformed configuration in the local frame of pose $\boldsymbol{X}_i$. We optimize (6.11) using a Levenberg-Marquardt method in GTSAM [6].

Since the deformation graph contains a subsampled version of the original mesh, after the optimization, we retrieve the location of the remaining vertices as in [192]. In particular, the positions of the vertices of the complete mesh are obtained as affine transformations of nodes in the deformation graph:

$$\widetilde{\boldsymbol{v}}_i = \sum_{j=1}^{m} s_j(\boldsymbol{v}_i)[\boldsymbol{R}^M_j(\boldsymbol{v}_i-\boldsymbol{g}_j)+\boldsymbol{t}^M_j] \tag{6.12}$$

where $\boldsymbol{v}_i$ indicates the original vertex positions and $\widetilde{\boldsymbol{v}}_i$ are the new deformed positions. The weights s_j are defined as

$$s_j(\boldsymbol{v}_i) = (1 - ||\boldsymbol{v}_i - \boldsymbol{g}_j||/d_{\max})^2 \tag{6.13}$$

and then normalized to sum to one. Here $d_{\max}$ is the distance to the $k+1$ nearest node as described in [192] (we set $k = 4$).

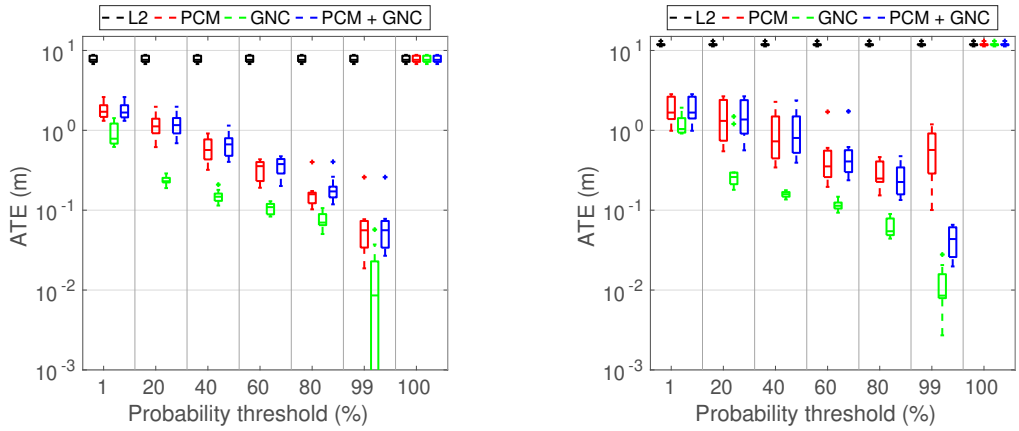
Note that the Kimera-Semantics mesh also includes semantic labels as an attribute for each node in the mesh, which remain untouched in the mesh deformation.

6.6 Offline Experiments

In this section, we perform offline evaluations of Kimera-Multi using photo-realistic simulations and real-world datasets. Our results show that Kimera-Multi provides robust and accurate estimation of trajectories and metric-semantic meshes, is efficient in terms of communication usage, and is flexible thanks to its modularity. The rest of this section is organized as follows. In Section 6.6.1, we analyze the robustness of Kimera-Multi in numerical experiments. In Section 6.6.2, we evaluate the quality of trajectory estimates and metric-semantic reconstruction in photo-realistic simulations and benchmarking datasets. Lastly, in Section 6.6.3, we demonstrate Kimera-Multi on two challenging real-world datasets collected by ground robots.

6.6.1 PGO Robustness Analysis

In this section, we evaluate different robust trajectory estimation techniques on synthetic datasets with varying ratios of outlier loop closures. Our results demonstrate the importance of robust initialization for multi-robot PGO. Furthermore, we show that alternative technique based on PCM [81] has low recall (*i.e.*, missing correct loop closures). Overall, we show that the proposed D-GNC method achieves the best performance, and is not sensitive to parameter tuning.



(a) Outlier ratio: 10%

(b) Outlier ratio: 70%

Figure 6-6: **Single-robot tests.** Comparisons between solvers on single-robot synthetic PGO problems across 10 Monte Carlo runs.

Single-robot experiments. To offer additional insights and contrast with the multi-robot analysis later, we first perform ablation studies on single-robot synthetic datasets. We simulate 2D PGO problems contaminated by outliers using the INTEL dataset [193]. To generate outlier loop closures, we randomly select pairs of non-adjacent poses in the original pose graph, and add relative measurements with uniformly random rotations and translations. For translations, we sample each coordinate uniformly at random within the domain $[-10, 10]$ m.

The following trajectory estimation techniques are compared: (1) L2: standard least squares optimization using Levenberg-Marquardt (LM), (2) PCM: outlier rejection with pairwise consistency maximization [81] using the approximate maximum clique solver [188] followed by LM, (3) GNC: graduated non-convexity [28], (4) PCM + GNC: PCM outlier rejection followed by GNC. Both LM and GNC are implemented in GTSAM [6]. All methods start from the odometry initial guess. Note that both PCM and GNC require the user to specify a confidence level (in the form of a probability threshold) that determines the maximum residual of inliers. We vary this probability threshold and compare different techniques across the entire spectrum.

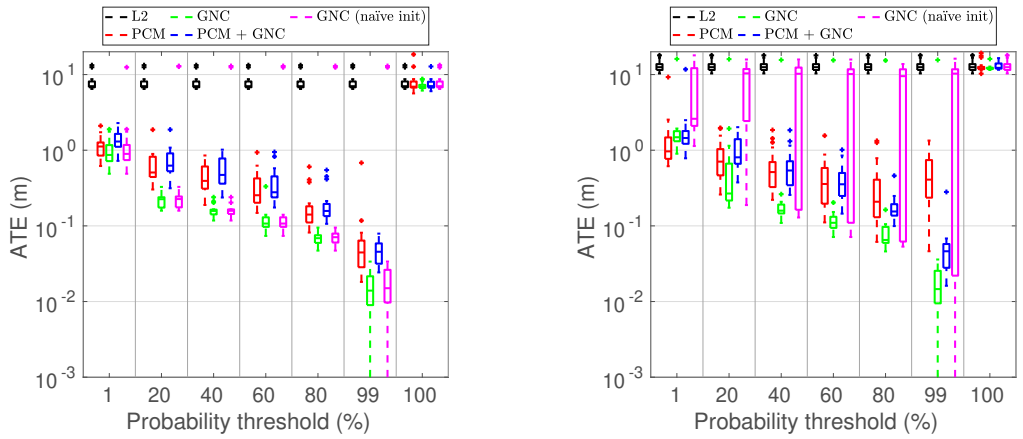
Figure 6-6 shows the absolute trajectory errors (ATE) with respect to the maximum likelihood estimate, computed using the outlier-free pose graph. Results are collected over 10 Monte Carlo runs. Standard LM optimization is not robust even

under 10% outlier loop closures (Figure 6-6a). In many cases, PCM tends to be overly conservative and reject inliers (due to approximate maximum clique search), which leads to an increase in the trajectory error. The same issue also negatively impacts the performance of PCM + GNC (blue), since rejected inliers cannot be recovered. On the other hand, GNC (green) achieves smaller error across the entire spectrum. Under 70% outliers (Figure 6-6b), PCM has larger errors especially at higher probability thresholds (*e.g.*, 99%), indicating that the method is unable to reject all outliers. In this case, applying subsequent GNC helps to improve the performance of PCM. However, also in this case, applying GNC alone consistently achieves the best performance over the entire range of probability thresholds. This result suggests that GNC should be the method of choice in single-robot PGO independent from the parameter tuning.

Multi-robot experiments. In multi-robot PGO, there is no longer an outlier-free initial guess (*i.e.*, odometry), which is crucial for the strong performance of GNC observed in the single-robot case. We investigate this issue in the next experiment, and demonstrate the robust initialization scheme proposed in Section 6.4.2 as an effective solution. Similar to the previous experiment, we use the INTEL dataset with the same outlier model described previously. The pose graph is divided into three segments with approximately equal lengths to simulate a three-robot collaborative SLAM scenario.

We compare two variants of GNC using different initial guesses. The first variant uses the proposed robust initialization scheme, and is labeled as "GNC" in Figure 6-7 (green). The second variant uses a naïve initialization formed using the local odometry of each robot and randomly sampled inter-robot loop closures between pairs of robots; see (6.4). This variant is labeled as "GNC (naïve init)" in Figure 6-7 (magenta). When PCM is used, we sample inter-robot loop closures from the inlier set returned by PCM. All problems are solved using a centralized implementation based on GTSAM [6]. Distributed experiments will be presented in the next section.

Figure 6-7 reports ATE results across 10 Monte Carlo runs. With 10% outlier loop closures (Figure 6-7a), it is less likely that the naïve initialization is affected by



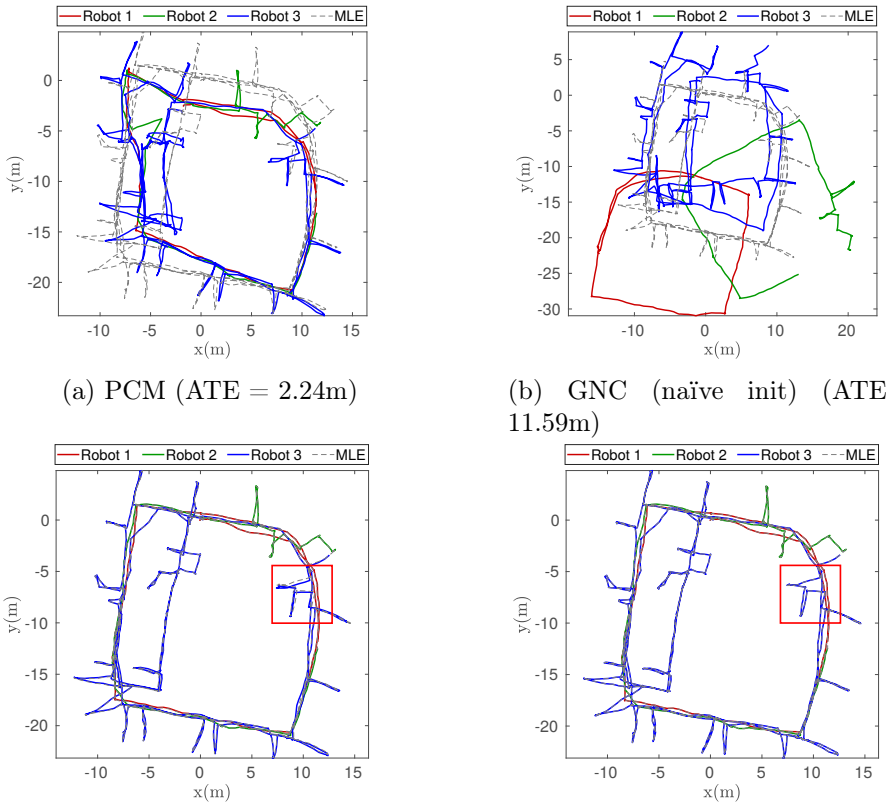
(a) Outlier ratio: 10%

(b) Outlier ratio: 70%

Figure 6-7: **Multi-robot tests.** Comparisons between solvers on three-robot synthetic PGO problems across 10 Monte Carlo runs.

outliers. Consequently, the two variants of GNC have similar performance in most cases, but naïve initialization still causes occasional failures (magenta outliers). The failure cases correspond to instances when the initial guess was accidentally built using an outlier loop closure. The problem caused by incorrect initialization becomes more evident under 70% outlier loop closures (Figure 6-7b), where naïve initialization fails in the majority of instances. This is because under 70% outliers, the naïve initial guess is almost always contaminated by wrong loop closures, which severely affects the performance of GNC. In comparison, using PCM helps to avoid catastrophic failures, but PCM still exhibits low recall as in the single-robot case. Finally, the proposed robust initialization effectively corrects the wrong initial guess, and applying GNC from the robust initialization (green) consistently outperforms other techniques.

To provide additional insights over the performance of different techniques, Figure 6-8 shows qualitative comparisons of final trajectory estimates on a random problem instance with 70% outliers. All techniques use the same probability threshold of 99%. Under this setting, PCM (Figure 6-8a) fails to reject all outlier loop closures. As a result, its solution is distorted when compared to the maximum likelihood estimate. When applying GNC from naïve initialization (Figure 6-8b), the method fails to recover any inlier loop closures due to incorrect initialization that causes the variable update to converge to wrong estimates. Figure 6-8c-6-8d show that applying either



(a) PCM (ATE = 2.24m)

(b) GNC (naïve init) (ATE = 11.59m)

(c) PCM + GNC (ATE = 0.09m)

(d) GNC (ATE = 0.003m)

Figure 6-8: Comparing final trajectory estimates of different techniques under 70% outlier loop closures. All methods use the same probability threshold of 99%.

PCM or robust initialization to correct the initial guess before applying GNC can effectively resolve the problem. Between this two approaches, however, our proposed robust initialization produces lower trajectory error, which can also be seen by comparing the trajectory estimates within the red box. This is because PCM incorrectly removes inlier loop closures during outlier rejection, which causes a loss of accuracy that cannot be recovered by GNC.

6.6.2 Evaluation in Simulation and Benchmarking Datasets

We evaluate Kimera-Multi in three photo-realistic simulation environments (Medfield, City, Camp), developed by the Army Research Laboratory Distributed and Collaborative Intelligent Systems and Technology (DCIST) Collaborative Research Alliance [194]. Figures 6-9 to 6-11 show the dense 3D mesh models built by Kimera-Multi in

Table 6.1: Absolute trajectory errors (ATE) in meters with respect to ground truth trajectories. For each dataset, we also report the total trajectory length (including all robots). L2: standard least squares optimization using LM; PCM: pairwise consistency maximization [81]; D-GNC: proposed distributed trajectory estimation method (using robust initialization); NI: naïve initialization; ES: early stopping. For reference, we also report the ATE of centralized GNC (colored in **gray**).

	Length [m]	L2	PCM	D-GNC (NI)	PCM + D-GNC	D-GNC	D-GNC (ES)	Centralized GNC
Medfield	2396	64.2	12.5	57.4	4.64	**3.92**	4.32	**3.88**
City	1213	3.58	1.57	0.91	1.08	0.85	**0.76**	**1.00**
Camp	1037	11.9	1.37	0.97	1.09	0.96	**0.75**	**1.33**
Vicon Room 1	211	1.17	1.00	0.34	0.45	0.35	**0.21**	**0.36**
Vicon Room 2	206	1.87	1.56	0.46	0.62	**0.47**	0.48	**0.43**
Machine Hall	466	1.92	1.76	0.48	0.70	**0.41**	0.49	**0.52**

these simulated environments. In addition, we also evaluate on three real-world environments (Vicon Room 1, Vicon Room 2, Machine Hall) from the EuRoc dataset [195]. Among all datasets, Machine Hall contains five sequences which are used to simulate collaborative SLAM with five robots. The simulation and Vicon Room datasets contain three sequences that are used to simulate a three-robot scenario. In our experiments in this section and Section 6.6.3, we run Kimera-Multi in a setting where robots are constantly in communication range, which means that inter-robot loop closures are established at the earliest possible time.

Trajectory estimation results. We first evaluate the accuracy of different distributed trajectory estimation techniques. In this experiment, we use Kimera-VIO to process raw sensor data, and it is thus hard to obtain accurate covariance information for all measurements. In our implementation, we use a fixed isotropic covariance for each residual in PGO, with a standard deviation of 0.01 rad for rotation and 0.1 m for translation. Moreover, we use a relatively conservative probability threshold of 50% for all robust estimation techniques. We compare the following distributed solvers: (1) L2: standard PGO (least squares optimization) using RBCD (Chapter 4), (2) PCM: outlier rejection with PCM [81], followed by RBCD, (3) D-GNC (NI): proposed D-GNC method starting from a naïve initial guess that combines local odometry of each robot with randomly sampled inter-robot loop closures between pairs of robots, (4) PCM + D-GNC: outlier rejection with PCM, followed by D-GNC,

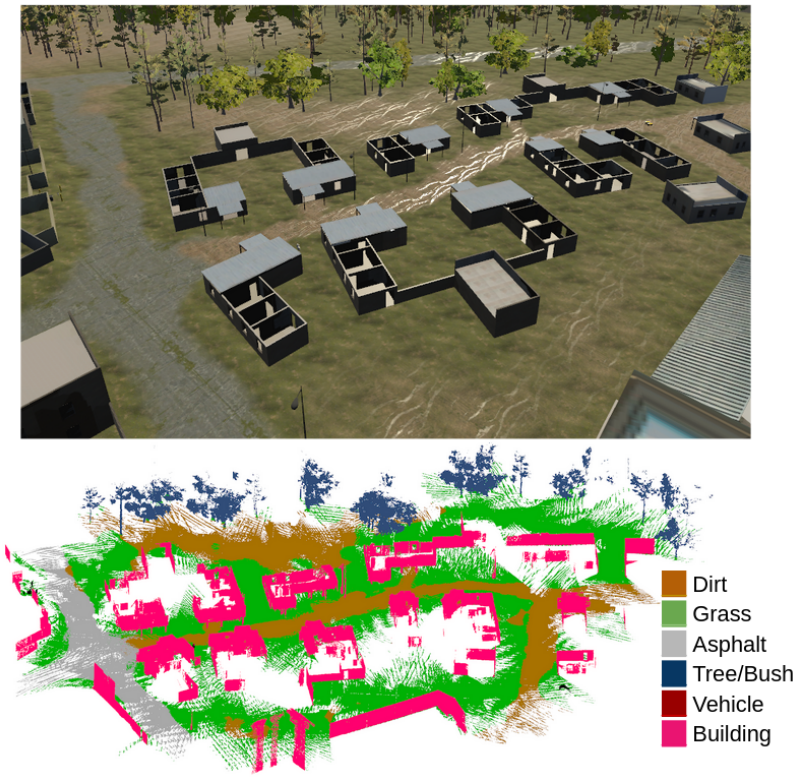


Figure 6-9: Dense metric-semantic 3D mesh model generated by Kimera-Multi with three robots in the simulated Camp scene.

(5) D-GNC: the proposed D-GNC method with robust initialization, (6) D-GNC (ES): an "early stopped" version of D-GNC that terminates after 50 total RBCD updates, (7) centralized GNC from GTSAM [6].

Table 6.1 reports the final ATE of each method when evaluated against the ground truth. Note that the total trajectory length varies significantly across datasets, which also causes ATE to vary. Due to the existence of outlier loop closures, standard least squares optimization (L2) gives large errors. PCM improves over the L2 results, but still yields large errors on a subset of datasets. The proposed D-GNC method achieves significantly lower trajectory errors on all datasets. Similar to the synthetic experiments (Section 6.6.1), we observe that applying GNC after PCM ("PCM + D-GNC" in the table) always leads to suboptimal performance compared to the proposed approach, due to the low recall of PCM. On the Medfield simulation, applying D-GNC from naïve initialization fails. In this case, the naïve initialization is wrong due to the selection of an outlier loop closure. This creates an error in the initial alignment of

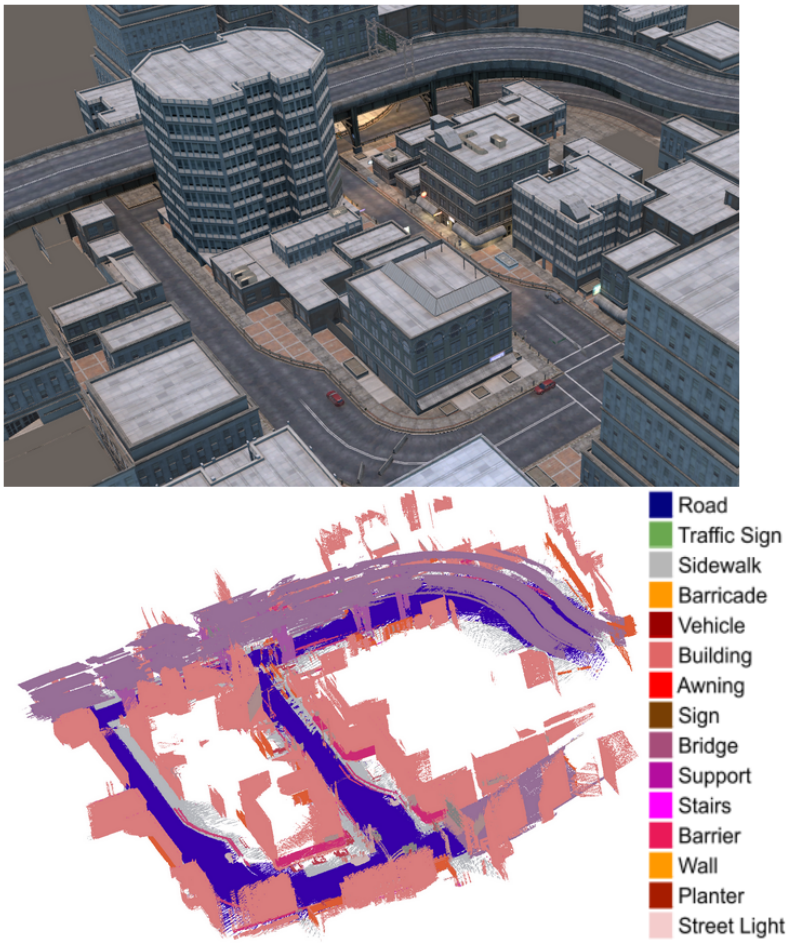


Figure 6-10: Dense metric-semantic 3D mesh model generated by Kimera-Multi with three robots in the simulated City scene.

robots' reference frames which D-GNC is unable to correct. Finally, we observe that on three of the datasets, applying early stopping (ES) leads to lower error compared to full optimization (distributed or centralized). In this experiment, estimation errors are computed with respect to the ground truth trajectories, which are in general different from the true (unknown) maximum likelihood estimate. In summary, the proposed D-GNC method achieves the best performance, and applying early stopping (ES) does not significantly affect the accuracy of trajectory estimation, which remains comparable to the centralized GNC.

Communication usage and solution runtime. In Table 6.2, we compare the communication usage of Kimera-Multi with two baseline centralized architectures that either transmit all images or keypoints. Data payloads used by Kimera-Multi are di-

Figure 6-11: Dense metric 3D mesh model generated by Kimera-Multi with three robots in the simulated Medfield scene.

vided into three parts: place recognition (exchanging bag-of-word vectors), geometric verification (exchanging keypoints and descriptors), and distributed PGO. The front-end (first two modules) consumes more communication than the back-end (distributed PGO). Overall, our results demonstrate that Kimera-Multi is communication-efficient. For instance, on the Vicon Room 2 dataset, our system achieves a communication reduction of 70% compared to the baseline centralized system that transmits all keypoints and descriptors. On the other hand, the system does not achieve equally significant communication reduction on the Machine Hall dataset. Compared to other datasets, the increased number of robots in Machine Hall results in more data transmission. In particular, the loose thresholds for loop closure detection lead to increased data transmission during the geometric verification (GV) stage.

In addition, Table 6.2 reports the runtime of D-GNC and also compares with the centralized solver (implemented in GTSAM [6]). Our method has reasonable runtime (approximately 10 seconds) for the smaller Vicon Room datasets. For the larger datasets, D-GNC requires more time for full convergence. Nevertheless, applying early stopping (ES) effectively keeps the runtime close to its centralized counterpart,

Table 6.2: Communication usage and solution runtime. The data payloads induced by Kimera-Multi are further divided into three modules: place recognition (PR) that exchanges bag-of-word vectors, geometric verification (GV) that transmits keypoints and feature descriptors, and distributed pose graph optimization (DPGO). Centralized communication and runtime are colored in **gray**.

Dataset	# Poses	# Edges	Communication [MB]						Runtime [sec]		
			PR	GV	DPGO	Total	Centralized (Images)	Centralized (Keypoints)	Distributed	Distributed (ES)	Centralized
Medfield	2918	3104	22.6	41.5	1.8	**65.9**	**2113**	**141**	29.2	**5.9**	**4.4**
City	3212	4173	16.2	44.5	8.8	**69.5**	**2326**	**155**	22.1	**4.5**	**3.2**
Camp	5088	5200	39.2	19.7	0.5	**59.4**	**3685**	**246**	43.2	**9.1**	**4.4**
Vicon Room 1	1693	2788	9.5	14.7	3.6	**27.8**	**1226**	**81.7**	8.9	**2.2**	**3.1**
Vicon Room 2	1738	2335	11.5	10.2	2.7	**24.4**	**1259**	**83.9**	11.7	**3.2**	**1.7**
Machine Hall	3261	5196	46.0	76.8	22.9	**145.7**	**2362**	**157**	20.5	**2.5**	**6.3**

without heavily compromising estimation accuracy.

Metric-Semantic Mesh Quality. We use the ground-truth point clouds available in the EuRoc Vicon Room 1 and 2 datasets, and the ground-truth mesh (and its semantic labels) available in the DCIST simulator to evaluate the accuracy of the 3D metric-semantic mesh built by Kimera-Semantics and the impact of the local mesh optimization (LMO). For evaluation, the estimated and ground-truth meshes are sampled with a uniform density of 10^3 points/m^2 as in [186]. The resulting semantically-labeled point clouds are then registered using the ICP [196] implementation in *Open3D* [197]. Then, we calculate the mean distance between each point in the ground-truth point cloud to its nearest neighbor in the estimated point cloud to obtain the metric accuracy of the 3D mesh. In addition, we evaluate the semantic reconstruction accuracy by calculating the percentage of correctly labeled points [186] relative to the ground truth using the correspondences given by ICP. Figure 6-12 and Figure 6-13 report the metric accuracy of the individual meshes constructed by each robot as well as the merged global mesh, and Table 6.3 shows the semantic reconstruction accuracy in the simulator (EuRoc does not provide ground-truth semantics). In general, the metric-semantic mesh accuracy improves after LMO for both individual and merged 3D meshes, demonstrating the effectiveness of LMO in conjunction with our distributed trajectory optimization. The dense metric-semantic meshes are shown in Figure 6-9 and Figure 6-10. In the case when semantic labels are unavailable, we are still able

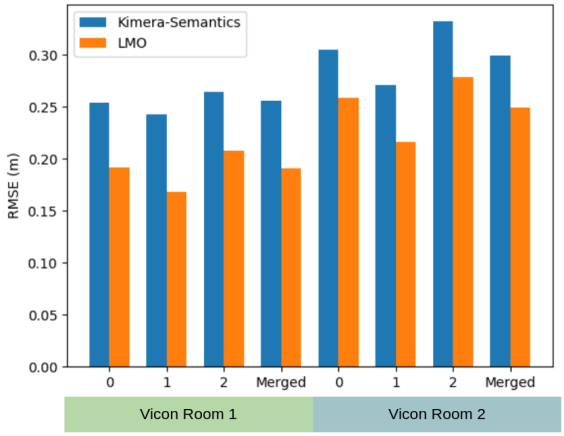


Figure 6-12: Metric reconstruction evaluation on the Euroc sequences. Mesh error (in meters) for the 3D meshes by Kimera-Semantics and Kimera-Multi's LMO.

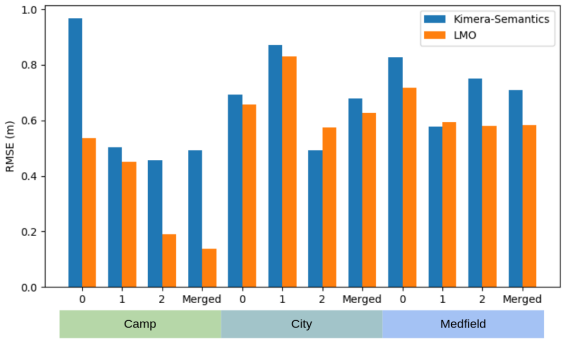


Figure 6-13: Metric reconstruction evaluation on the Camp, City, and Medfield simulator datasets. Mesh error (in meters) for the 3D meshes by Kimera-Semantics and Kimera-Multi's LMO.

to generate the mesh, colored by the RGB image colors, as shown in Figure 6-11 for the experiment in the simulator portraying the Medfield scene.

6.6.3 Evaluation in Outdoor Datasets

Experimental Setup. We demonstrate Kimera-Multi on two challenging outdoor datasets, collected using a Clearpath Jackal UGV equipped with a forward-facing RealSense D435i RGBD Camera and IMU. The first dataset was collected at the Medfield State Hospital, Massachusetts, USA (Figure 6-1). Three sets of trajectories were recorded, with the longest trajectory being 860 meters in length. The second dataset was collected around the Ray and Maria Stata Center at MIT (Figure 6-14), and also includes three different trajectories with each trajectory being over 500 meters in length. In both Figure 6-1 and Figure 6-14, the red, orange, and

Table 6.3: Semantic reconstruction evaluation. Semantic labels accuracy before and after correction by LMO in the DCIST simulator.

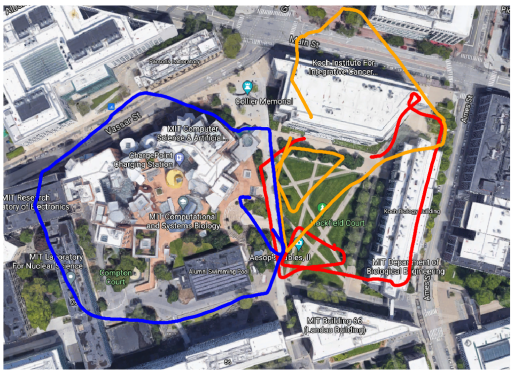
Dataset	Robot ID	Kimera-Semantics (%)	LMO (%)
Camp	0	81.6	**96.2**
	1	92.8	**98.1**
	2	82.8	**96.1**
	Merged	79.4	**95.2**
City	0	77.1	**77.7**
	1	80.7	**83.1**
	2	**71.4**	70.6
	Merged	76.1	**78.8**

blue trajectories correspond to robots with ID 0, 1, and 2, respectively. Both sets of experiments are challenging and include many similar-looking scenes that induce spurious loop closures. The results presented below are obtained by replaying the data sequences at reduced speed on a single desktop computer.

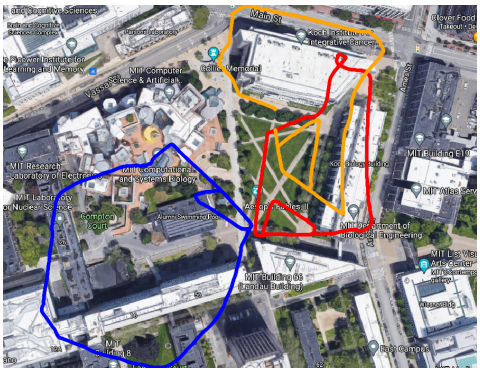
Results and Discussions. Table 6.4 reports statistics about loop closures on the outdoor datasets. Specifically, for each pair of robots, we report the number of loop closures accepted by D-GNC over the total number of detected loop closures (including outliers). Diagonal entries in the table correspond to intra-robot loop closures. Both datasets contain many outlier loop closures, which are successfully rejected by D-GNC. Compared to Medfield, the Stata dataset contains significantly less inter-robot loop closures, which makes distributed PGO particularly challenging.

In order to evaluate estimation accuracy in the absence of ground truth trajectories, we measure end-to-end errors as in [198]. In particular, we design each individual robot trajectory to start and finish at the same place, and then compute the final end-to-end position errors. The end-to-end error is not equivalent to the ATE, but still provides useful information about the final estimation drift on each trajectory. Table 6.5 compares the end-to-end errors of Kimera-VIO, Kimera-Multi (using D-GNC to estimate trajectories), and centralized result (solved using GNC in GTSAM [6]). To complement the quantitative result, we also provide qualitative visualizations of the optimized trajectories and meshes in Figure 6-1 and Figure 6-14.

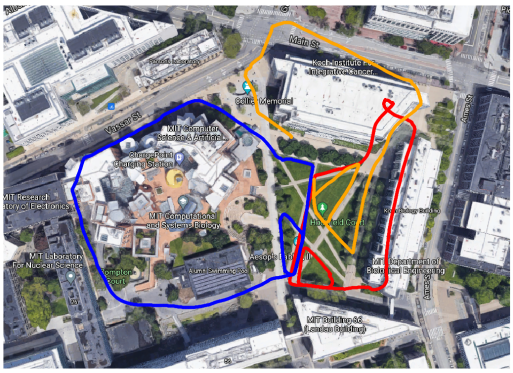
On the Medfield dataset (Figure 6-1), Kimera-VIO accumulates a drift of approximately 15-25 m on each trajectory sequence. We note that the drift is mostly in the



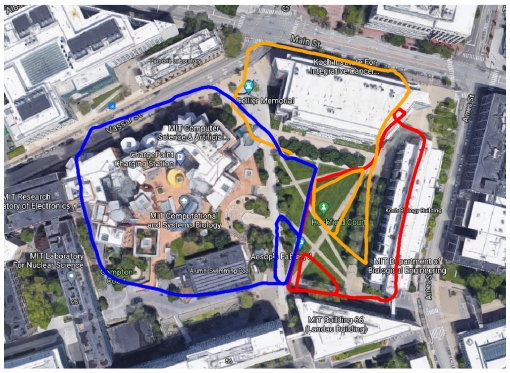
(a) Kimera-VIO



(b) Kimera-Multi (D-GNC with approximate variable updates)



(c) Kimera-Multi (D-GNC with full variable updates)



(d) Centralized GNC

Figure 6-14: **Stata experiment.** (a) Trajectory estimate from Kimera-VIO. (b) Trajectory estimate produced by Kimera-Multi, using D-GNC with the default approximate variable updates. (c) Trajectory estimate produced by Kimera-Multi, using D-GNC with full variable updates. (d) Trajectory estimate produced by centralized GNC.

vertical direction, hence only partially visible in Figure 6-1a. Through loop closures and robust distributed PGO, Kimera-Multi significantly reduces the error and furthermore achieves the same performance as the centralized solver as shown in Table 6.5. In this case, the global pose graph has 15650 poses in total (including all robots). D-GNC uses a total of 100 RBCD iterations which takes 53 seconds. Further runtime reduction may be achieved by decreasing the rate at which keyframes are created. In summary, our trajectory estimation results together with the final optimized mesh shown in Figure 6-1c demonstrate the effectiveness of the proposed system.

In comparison, the Stata dataset (Figure 6-14) is more challenging, partially due to the lack of enough inter-robot loop closures (Table 6.4b). Kimera-VIO accumu-

Figure 6-15: **Stata experiment.** Optimized mesh produced by Kimera-Multi corresponding to trajectory estimate shown in Figure 6-14c.

Table 6.4: Loop closure statistics on outdoor datasets. For each pair of robots, we show the number of loop closures accepted by D-GNC over the total number of putative loop closures (including outliers). Diagonal entries correspond to intra-robot loop closures.

(a) Medfield experiment

	Robot 0	Robot 1	Robot 2
Robot 0	1/1	11/41	27/53
Robot 1		79/114	340/707
Robot 2			172/182

(b) Stata experiment

	Robot 0	Robot 1	Robot 2
Robot 0	391/416	1/2	1/1
Robot 1		217/271	0/0
Robot 2			57/76

lates higher drifts as shown in Figure 6-14a and Table 6.5. Figure 6-14b shows the Kimera-Multi trajectory estimates produced using the default settings of D-GNC (see Algorithm 6.1). In this case, the global pose graph has 11184 poses in total. D-GNC uses 120 RBCD iterations which takes 50 seconds. We observe that while the orange and red trajectory estimates are qualitatively correct, the blue trajectory is not correctly aligned in the global frame. This is because with the approximate variable updates of D-GNC (presented in Section 6.4.3), the only inter-robot loop closure with the blue trajectory is rejected. Additionally, with fewer inter-robot loop closures, RBCD generally converges at a slower rate. To resolve this issue, we increase the number of RBCD iterations within each variable update, hence making D-GNC more similar to the centralized GNC algorithm. With this change, D-GNC uses a

Table 6.5: Trajectory lengths and end-to-end errors in meters on outdoor datasets.

Dataset	Robot ID	Length [m]	Kimera-VIO	Kimera-Multi	Centralized
Medfield	0	600	18.74	0.01	0.01
	1	860	14.84	0.13	0.13
	2	728	24.55	0.09	0.09
Stata	0	515	49.02	0.03	0.01
	1	570	24.19	33.13	21.56
	2	610	29.35	1.26	1.17

total of 2000 RBCD iterations which takes 14 minutes. However, the final trajectory estimates (Figure 6-14c) are significantly improved and are close to centralized GNC (Figure 6-14d). The corresponding end-to-end errors are also close to centralized GNC as shown in Table 6.5. Figure 6-15 shows the optimized mesh produced by Kimera-Multi corresponding to the trajectory estimates shown in Figure 6-14c. In this experiment, the difficulty faced by Kimera-Multi is primarily due to the lack of inter-robot loop closures (Table 6.4b).

6.7 Large-Scale Field Experiments

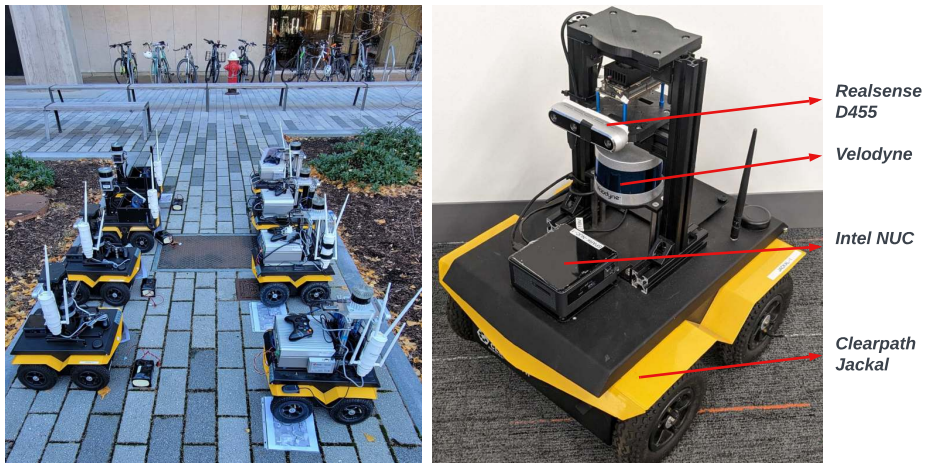


Figure 6-16: The Jackal robots used for the experiments.

In this section, we present results from large-scale field experiments of Kimera-Multi on the MIT campus with a fleet of Clearpath Jackal rovers equipped with a Realsense

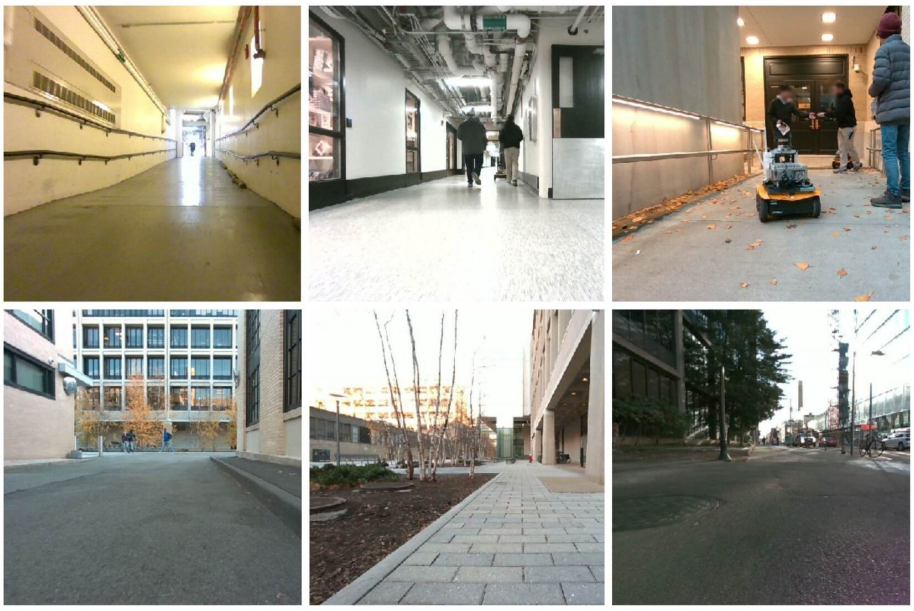


Figure 6-17: Snapshots from the Campus-Hybrid dataset.

D455 RGB-D Camera and an Ouster or Velodyne 3D lidar (Figure 6-16). Each robot has an Intel NUC computer with an Intel i7 4.70 GHz processor, and communicates with the other robots via a 2.4 GHz wireless network. Each robot was tele-operated by a human operator. During the experiment, there were often humans and vehicles moving around the robots, and the robots traverse through a variety of both indoor and outdoor environments (Figure 6-17).

6.7.1 Datasets

We compiled three real-world datasets based on the data recorded. The first dataset is the Campus-Outdoor dataset (Figure 6-19a), which consists of six Jackal rovers traversing a total of 6044 meters over a duration of around 20 minutes on the MIT campus. Except for one of the robots traversing a crowded building, most of the dataset consists of outdoor campus and urban scenes. The second dataset is the Campus-Tunnels dataset (Figure 6-19b), which consists of eight Jackal rovers traversing a total of 6753 meters over a duration of almost 30 minutes in the tunnels underneath the university campus. The trajectories are entirely indoors and consist mostly of long homogeneous tunnels and corridors. The last dataset is the Campus-Hybrid dataset (Figure 6-18),

which consists of eight Jackal rovers traversing a total of 7785 meters over a duration of almost 30 minutes both on and below the university campus. This dataset is also multi-level, with some trajectories crossing each other on different levels.

The reference trajectory we use for evaluation is generated by first building a reference point cloud map as shown in the background of Figure 6-18 using lidar-based SLAM [3, 199], with additional total station measurements for the indoor portions of the datasets and differential GPS measurements throughout the outdoor areas in and around the university campus. Once we have the reference point cloud map, we obtain the reference trajectories using LOCUS 2.0 [199] by matching the lidar point cloud for each robot at each time-step against the reference point cloud map. All datasets and reference trajectories are publicly available.[8]

6.7.2 Experimental Setup

The experiments in this section are obtained by playing back the datasets on robots in real-time. All agents communicate with each other via a 2.4 GHz wireless network. On top of the real communication, we simulate different types of disconnections (discussed in the next paragraph) by controlling connectivity in the remote topic manager module (Section 6.2). Apart from the simulated communication disruptions, other aspects of the experiments such as available onboard compute and the rate at which the input is received by Kimera-Multi are equivalent to the live experiments.

Communication Scenarios. We evaluate our system under four simulated communication scenarios (implemented in the remote topic manager):[9] (i) Full: all nodes are connected at all times; (ii) Random: each node is disconnected randomly three times during the mission, and each disconnection lasts 90 seconds. This scenario tests the resilience of our system to situations in which some robots experience temporary failures and go offline for some time; (iii) Distance: connection is established between nodes within a distance threshold. Note that under this setup, there can be

[8] https://github.com/MIT-SPARK/Kimera-Multi-Data

[9] The simulation applies to all robots, and optionally to the base station (placed at the origin) when running the centralized baseline. For the purpose of computing the trajectory error, for each dataset we start the simulation after at least two robots are initialized in the global frame.

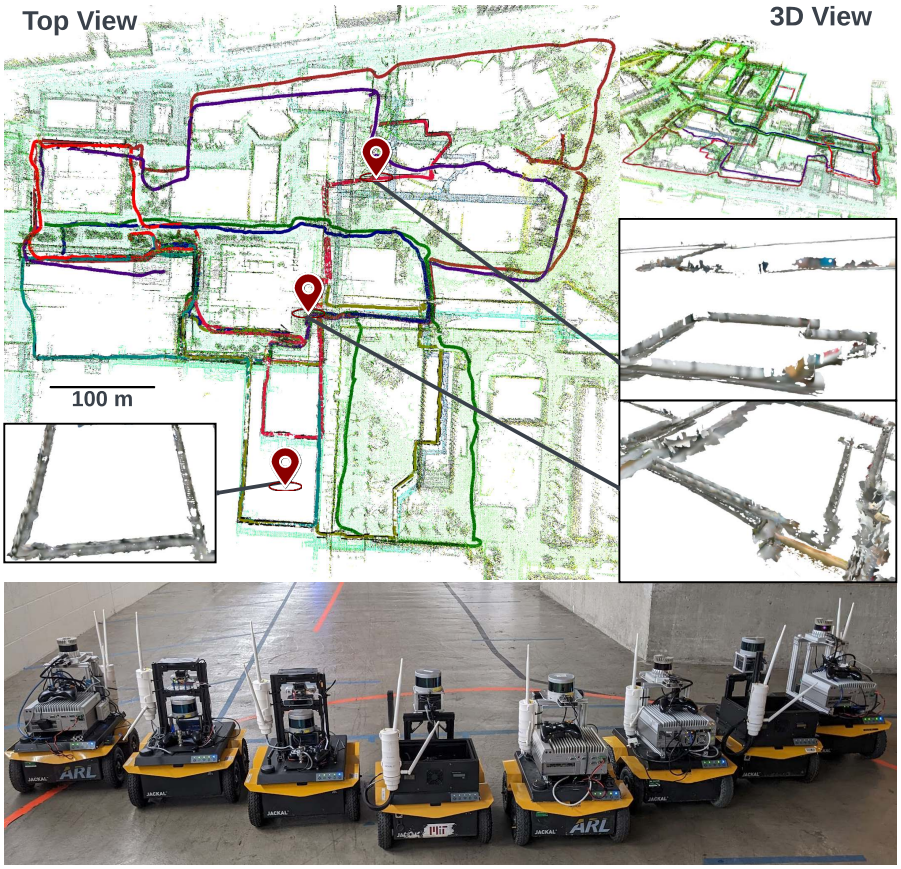
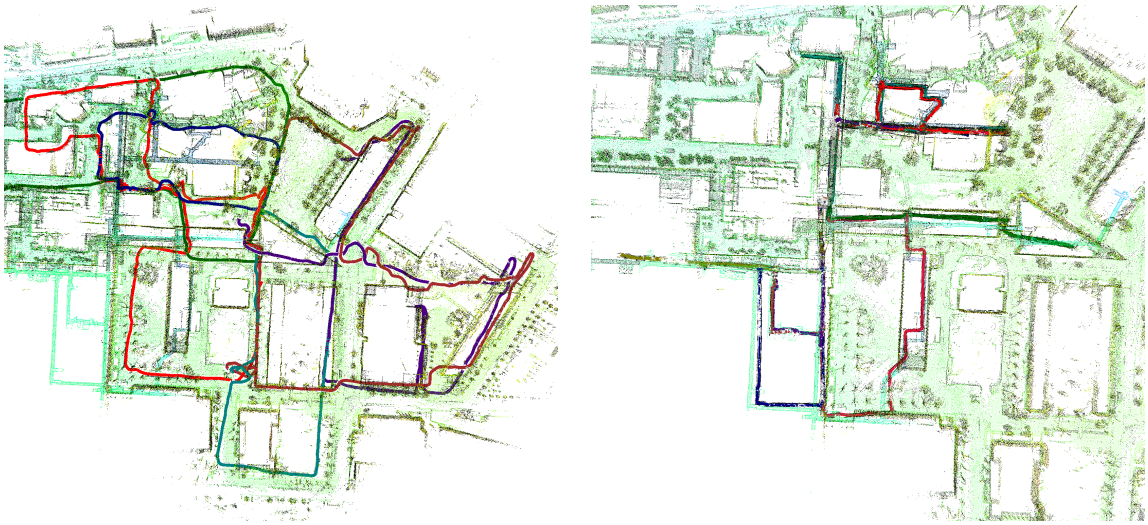


Figure 6-18: Kimera-Multi trajectories and meshes overlaid on top of the reference point cloud map from the Campus-Hybrid experiment where 8 robots traverse indoor-outdoor scenes covering a total of 7785m.

multiple clusters of connected nodes at any given time. In our experiments, we set the distance threshold to 75m, which we observe to create interesting and challenging network topologies on all datasets. We assume robots in each connected component can help route information to others, so that all robots in the same cluster can communicate with each other; (iv) Base: nodes can communicate only within a certain radius of the base station (75m and same as Distance). This setting is the closest to the communication setup observed in our live experiments.

Centralized Baseline. We implement a centralized baseline system that detects all inter-robot loop closures at a base station. The back-end subscribes to the pose graphs and the loop closures from the robots, and optimizes the full problem using GNC with Levenberg-Marquardt as implemented with GTSAM [6]. The loop closure



(a) Campus-Outdoor (b) Campus-Tunnels

Figure 6-19: Estimated trajectories, estimated meshes, and reference point cloud for the Campus-Outdoor (6 robots, 6044m) and Campus-Tunnels (8 robots, 6753m) datasets.

Table 6.6: Summary of front-end and cack-end statistics for Kimera-Multi (Dist) and centralized baseline (Cent).

		Front-end			Back-end						
		Loop Closures			Size	Iterations		Optimization Time		Metric Error	
		# Matches	# Loops	% Inliers	# Poses	# Max	# Median	Max(s)	Avg(s)	ATE(m)	AME(m)
Dist	Tunnels	12397 ± 74	10620 ± 70	20.8 ± 0.3	2832 ± 6	1083 ± 23	571 ± 68	285.55 ± 31.74	139.52 ± 15.52	4.48 ± 0.29	2.62 ± 0.07
	Hybrid	2999 ± 136	1252 ± 30	26.9 ± 0.2	3201 ± 6	1190 ± 62	698 ± 133	336.98 ± 26.25	174.05 ± 14.97	7.71 ± 0.78	1.9 ± 0.26
	Outdoor	714 ± 19	172 ± 10	36.5 ± 2.3	2383 ± 1	735 ± 327	344 ± 31	196.9 ± 43.11	100.92 ± 11.46	12.4 ± 1.92	4.83 ± 1.35
Cent	Tunnels	12432 ± 70	8247 ± 188	19.8 ± 0.4	3053 ± 4	-	-	11.51 ± 1.05	3.24 ± 0.28	4.38 ± 0.21	2.58 ± 0.12
	Hybrid	3052 ± 45	1302 ± 18	24.7 ± 0.5	3508 ± 1	-	-	11.44 ± 0.21	1.65 ± 0.16	5.83 ± 0.16	1.58 ± 0.05
	Outdoor	732 ± 6	182 ± 5	37.4 ± 1.0	2647 ± 0	-	-	11.49 ± 2.94	0.55 ± 0.12	9.38 ± 0.31	4.99 ± 0.48

parameters, GNC parameters, along with other related parameters like BoW vector downsampling and pose graph coarsening are consistent with the distributed setup. Additionally, details such as the initial reference frame alignment were implemented to be congruent to its distributed counterpart. For our experiments, the base station is ran on a laptop with an Intel i9 2.40 GHz processor.

6.7.3 Real-time Evaluation Under Unreliable Communication

In the following, we first discuss the key statistics from the front-end, the back-end, and the communication system. Then, we discuss in depth the performance for each of the data sequences under each communication scenario.

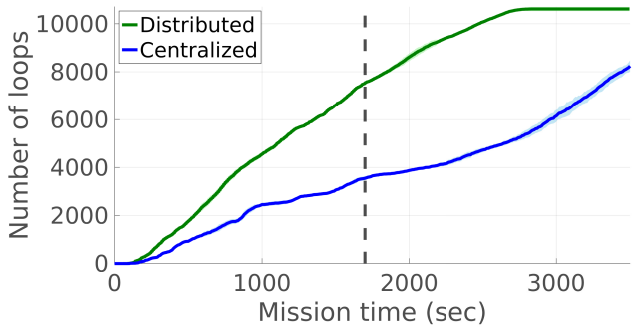


Figure 6-20: Number of detected loop closures on the Campus-Tunnels dataset under the Full communication scenario. Vertical dashed line denotes the time when all robots finish exploration.

Front-end Statistics. We present a summary of front-end statistics in Table 6.6. The number of BoW matches, detected loop closures, and the percentage of inlier loop closures are summarized over 3 trials under the Full communication scenario. A loop closure is classified as an inlier if the corresponding relative transformation is within 10 cm in translation and 10 degrees in rotation compared to the ground truth. We only present the summary from the Full communication scenario. Results under other scenarios are quantitatively similar.

Significantly more loop closures are detected on the indoor Campus-Tunnels dataset as there are smaller viewpoint variations in narrow corridors compared to outdoor areas that are generally more spacious. In general, the distributed system and the centralized baseline detect similar number of loop closures with similar inlier percentage. On the Campus-Tunnels dataset, however, the distributed front-end detects significantly more loop closures than the centralized baseline. This is because we terminate the experiment at 3500 seconds (more than twice the duration of the actual dataset), and the centralized system has not yet finished processing all the loop closures; see Figure 6-20. Lastly, we note that there is a significant number of outlier loop closures detected across all datasets, which highlights the importance of implementing an outlier-robust back-end.

Back-end Statistics. Table 6.6 also presents the statistics from the distributed and centralized back-end. In particular, the following metrics are reported: (1) the number of poses in the PGO problem, (2) the maximum and median number of iter-

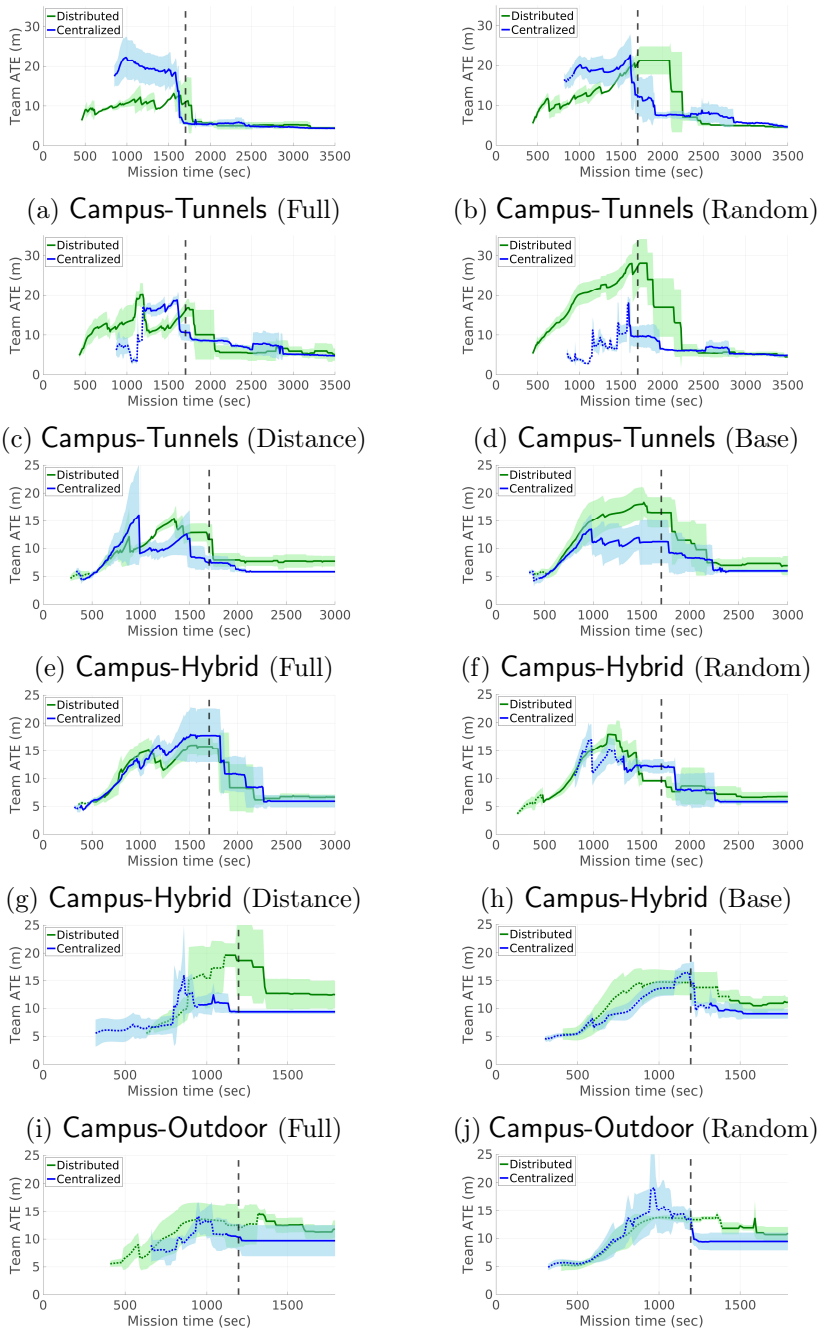
Table 6.7: Summary of communication statistics.

	Delay		Bandwidth		Packet Drop
	Max(s)	Avg(s)	Max(MB/s)	Avg(MB/s)	% Drop
Tunnels	7.71 ± 1.65	0.10 ± 9e-3	2.10 ± 0.02	0.55 ± 0.04	45.55 ± 7.89
Hybrid	4.17 ± 0.98	0.05 ± 3e-3	0.41 ± 0.02	0.12 ± 4e-3	21.50 ± 5.41
Outdoor	1.53 ± 0.36	0.05 ± 3e-3	0.10 ± 0.01	0.01 ± 2e-3	20.28 ± 5.72

ations taken by the distributed back-end, (3) the maximum and average optimization time, (4) the final absolute trajectory error (ATE), and (5) the final average map error (AME). Among these metrics, the ATE is defined as the root-mean-square error between the estimated and reference trajectories, and the AME is defined as the average mesh-vertex-to-point distance between the estimated mesh and the reference point cloud map. All metrics are computed by summarizing the results over 3 trials taken from the Full communication scenario.

For all three datasets, the distributed back-end achieves comparable but less accurate results compared to the centralized back-end, due to the fact that the distributed solver only solves GNC approximately. The distributed back-end also has a much higher optimization time, mainly because of the large number of iterations required by distributed PGO and the communication overhead accumulated across iterations. We remark that Table 6.6 compares the centralized and distributed system under Full communication; later in this section we discuss how different communication scenarios create a more interesting trade-off between centralized and distributed architectures.

Communication Statistics. Table 6.7 presents the communication profile of Kimera-Multi. The results are averaged over all pairs of robots from 3 trials under the Full communication scenario. During the experiments, there are sometimes delays of up to 7 seconds and a substantial amount of dropped packets (up to 45%), which demonstrates the challenges of using real-world wireless communication. The other three communication scenarios introduce even more challenges in the form of long periods of disconnections. Overall, our results demonstrate that Kimera-Multi is resilient to imperfect communication with large delays and unreliable packet delivery.



(a) Campus-Tunnels (Full) (b) Campus-Tunnels (Random)

(c) Campus-Tunnels (Distance) (d) Campus-Tunnels (Base)

(e) Campus-Hybrid (Full) (f) Campus-Hybrid (Random)

(g) Campus-Hybrid (Distance) (h) Campus-Hybrid (Base)

(i) Campus-Outdoor (Full) (j) Campus-Outdoor (Random)

(k) Campus-Outdoor (Distance) (l) Campus-Outdoor (Base)

Figure 6-21: ATE evaluation under different communication scenarios. Each row shows results on a single dataset, and each column corresponds to a communication scenario. Dashed line indicates that only a subset of robots is estimated, whereas solid line indicates that all robots are estimated.

Evolution of ATE. In Figure 6-21, we visualize the evolution of the ATE on the Campus-Tunnels, Campus-Hybrid, and Campus-Outdoor datasets under all four com-

munication scenarios. Each plot includes the ATE of both the distributed system and the centralized baseline, where the line represents the average ATE over 3 trials and the shaded area shows one standard deviation. The dashed portion of each line indicates that only a subset of robots is initialized in the global frame at that time, and consequently the ATE is only computed over that subset of robots. The line turns solid as soon as all robots are initialized in the global frame, at which point the ATE is computed over all robots. Finally, the vertical dashed line in each plot denotes the end of the mission (*i.e.* all robots finish mapping.)

The Campus-Tunnels dataset represents a scenario with many loop closures (see Table 6.6). For the Full communication scenario (Figure 6-21a), the distributed system is able to detect enough loop closures to initialize all robots in the global frame sooner and also achieves a lower ATE in the earlier portions of the data sequence. This is mostly due to the faster processing of the distributed front-end, which parallelizes loop closure detection across robots. However, the distributed back-end is slower than the centralized back-end, and the centralized system converges quickly after most of the loop closures are detected. At steady state, the performance of the centralized and distributed systems are similar. For the Random scenario (Figure 6-21b), we observe that random disconnects in the communication leads to a delayed decrease in ATE for both the distributed and centralized systems. However, the effect is more significant for the distributed system as the distributed back-end is interrupted by disconnections. The centralized system behaves similarly for the Distance and Base scenarios, but the distributed system performs better under the Distance scenario (Figure 6-21c) compared to the Base scenario (Figure 6-21d). This is because the distributed system is able to both detect loop closures and perform distributed PGO while being disconnected from the base station (as shown by the drop in ATE around 1300 seconds in Figure 6-21c).

In the Campus-Hybrid dataset, we again see the advantage of the distributed front-end as the distributed system stabilizes faster than the centralized baseline for the Full scenario (Figure 6-21e). For the Random scenario (Figure 6-21f), the overall ATE is increased for the distributed system due to delays caused by random disconnects.

The distributed system under the Distance (Figure 6-21g) and Base (Figure 6-21h) scenarios has similar performance to the centralized system throughout the mission, but the distributed system under the Distance scenario is able to maintain lower and more stable ATE between 1200 to 1800 seconds due to its flexibility to detect loop closures and perform PGO while being disconnected from the base station.

On the Campus-Outdoor dataset, the distributed front-end has little advantage due to the much smaller number of loop closures (see Table 6.6). This dataset is also challenging for the distributed back-end, as the sparse loop closures make the underlying optimization problem poorly conditioned and the distributed solver struggles to reach a good precision solution.

In summary, the distributed front-end is able to speed up loop closure detection by distributing the computational load across the robots; on the other hand, the distributed back-end requires substantially more time to converge compared to its centralized counterpart, but in some cases, it provides extra flexibility by enabling clusters of robots to optimize their trajectories while being away from the base station. This translates into reduced ATE in some portions of the trajectories (*e.g.* Figure 6-21a, 6-21b, and 6-21g).

6.7.4 Parameter Sensitivity

In this section, we present a detailed analysis of three key parameters that are observed to directly impact the accuracy of our system. These include two parameters related to our distributed solver: the gradient norm threshold ϵ_g that controls the precision when robots solve their local optimization problems, and the relative change threshold ϵ_{rel} that determines when to terminate distributed optimization. Intuitively, setting smaller values for these thresholds enable distributed PGO to obtain more accurate results at the expense of increasing iterations and runtime. The last parameter we analyze is the distance threshold d used to coarsen the pose graph. Recall from Section 6.3 that our system aggregates nearby pose graph nodes within this distance threshold. Thus, larger values of d yield a smaller and coarser pose graph. As we vary each parameter, we fix the remaining parameters to their nominal values ($\epsilon_g = 0.1$,

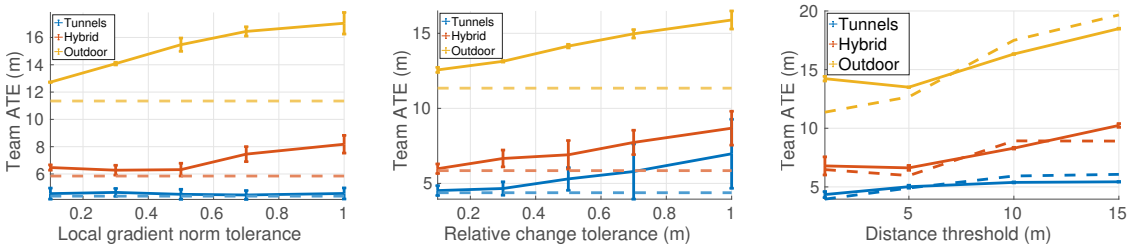
$\epsilon_{\text{rel}} = 0.2$m, and $d = 2$m).

Figure 6-22a-6-22b show the impact on ATE as we vary the PGO convergence parameters ϵ_g and ϵ_{rel}. For each dataset, we also show the reference ATE (constant dashed line) obtained from a centralized solver that solves GNC to full convergence. As expected, using smaller values (*i.e.* tighter termination conditions) helps the distributed back-end achieve similar accuracy as the reference solution. However, smaller values also lead to increased number of iterations; for example, decreasing ϵ_{rel} from 1.0 to 0.1 leads to 51%-180% more iterations across the three datasets. Larger values lead to worse solutions as optimization terminates before correcting all errors. Among the three datasets, Campus-Outdoor is the most sensitive to parameter changes due to poor conditioning caused by sparse loop closures. Meanwhile, Campus-Tunnels is less sensitive to the choice of ϵ_g, again due to the abundance of loop closures (Table 6.6) that helps robots improve their estimates even with a loose convergence parameter.

Figure 6-22c shows the effect of changing the distance threshold d for pose graph coarsening. The dashed lines again show the ATEs achieved by the centralized solver. Note that the dashed line is no longer constant because each value of d corresponds to a different PGO problem. Generally, decreasing d (*i.e.* increasing the pose graph resolution) yields better ATE. On the other hand, larger values of d significantly improve the efficiency of the back-end: across the three datasets, increasing d from 1m to 5m decreases the number of iterations by 38%-54%. We observe slightly different trends across datasets, where Campus-Outdoor is once again more sensitive to parameter changes. Lastly, on the Campus-Outdoor dataset we also observe a small increase in ATE towards smaller value of d. This is due to the fact that as d decreases, the pose graph becomes larger and the distributed back-end requires more iterations to achieve better accuracy.

6.7.5 Live Results and Discussions

In this section, we discuss key lessons learned from live field tests of Kimera-Multi and quantitative results. We summarize the capabilities and limitations of our system,



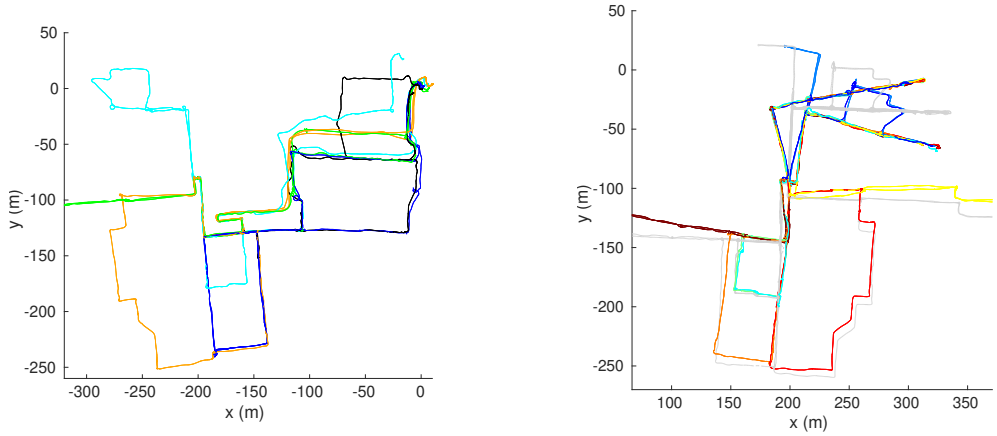
(a) ϵ_g (b) ϵ_{rel} (c) d

Figure 6-22: Effect on final ATE by varying key parameters: (a) local gradient norm threshold ϵ_g, (b) relative change tolerance threshold ϵ_{rel}, and (c) pose graph coarsening threshold d.

and present challenges and future work towards the resilient deployment of distributed multi-robot SLAM systems.

Resilience to Real-World Failures. Failures can happen in unexpected ways during real-world deployments. Kimera-Multi is resilient to some of these failure cases, and maintains its core CSLAM capability even if a subset of the robots experience total failures. Figure 6-23a shows a representative field test result, in which 8 robots initially explored a mixed indoor and outdoor scene similar to the Campus-Hybrid dataset, but 3 robots experienced hardware failures (2 robots ran out of battery and 1 robot's camera went offline) during the mission. Our system was able to adapt to the situation and obtain reasonable trajectory estimates for the remaining 5 robots. In general, resilience to unexpected failures is crucial for the reliable deployment of CSLAM systems.

Distributed Front-End (Loop Closure Detection). Oftentimes, the accuracy of trajectory estimation depends crucially on detecting sufficient loop closures. As shown in our experiments, by parallelizing loop closure detection among robots, the distributed front-end in Kimera-Multi is able to detect loop closures faster than the centralized baseline. Nevertheless, our front-end is subject to two limitations. First, the BoW-based place recognition is sensitive to the type of scenes, and often detects too many matches indoor and not enough outdoor (despite having trained the BoW vocabulary using both indoor and outdoor data). This makes setting parameters such as the similarity threshold α a challenging task, and using incorrect values could lead



(a) Hybrid (5 out of 8 robots)

(b) Tunnels (8 robots)

Figure 6-23: Trajectory estimates from example live experiments.

to significantly degraded performance.

Second, Kimera-Multi only performs visual loop closure detection, which is sensitive to changes in the viewpoint. This negatively impacts the performance on the Campus-Outdoor dataset, where some robots visit the same locations in opposite directions (*e.g.* right part of Figure 6-19a), and consequently no loop closure is detected and no trajectory correction occurs. View dependence is a well-known issue of vision-based loop closure methods. Further improvement in this module or incorporation of other approaches (*e.g.* hierarchical and object-based loop closures [200] and utilizing additional hardware such as ultra-wideband sensors) would be needed for Kimera-Multi to be resilient to different missions and trajectory plans.

Distributed Back-End (Pose Graph Optimization). Our results demonstrate that our back-end based on distributed GNC (Section 6.4) is able to achieve comparable accuracy as a centralized solver, and offers additional flexibility by enabling optimization within clusters of connected robots. The latest large-scale experiment with 8 robots shows that the accuracy of distributed GNC comes at the cost of increased runtime (Table 6.6). This is because on these larger problems, the distributed PGO solver requires more iterations to solve the intermediate optimization problems within GNC. Using a loose termination condition in the distributed solver could cause optimization to stop before correcting all errors, or even cause the rejec-

tion of critical inlier loop closures. Figure 6-23b shows an example failure case from a live field test for Campus-Tunnels, where we used a loose relative change threshold of $\epsilon_{\text{rel}} = 1\text{m}$. The result showed a significant error between the estimated trajectories (colored based on robots) and reference solution (colored in gray) on the top right part of the map. The worse performance is also consistent with the parameter sensitivity analysis presented in Section 6.7.4. In general, striking a good balance between optimization time and estimation accuracy can be difficult, and more work is needed to further improve the speed and scalability of distributed back-ends.

Communication. The communication module in Kimera-Multi is able to handle realistic network conditions with delays and message drops. However, in scenarios such as the Campus-Tunnels dataset where there are many loop closures, the front-end sometimes uses most of the available bandwidth for exchanging visual keypoints and descriptors, causing network congestion that interferes with the distributed back-end. To address this issue, recent methods that prioritize and impose a communication budget on inter-robot loop closure detection [3, 41, 85] would be helpful. Another potential solution is to implement *system-level* prioritization that allocates communication resources to different modules based on their current importance and the network condition.

6.8 Conclusion

This chapter presented Kimera-Multi, *a distributed multi-robot system for robust and dense metric-semantic SLAM.* Our system advances state-of-the-art multi-robot perception by estimating 3D mesh models that capture both dense geometry and semantic information of the environment. Kimera-Multi is *fully distributed*: each robot performs independent navigation, using Kimera to estimate local trajectories and meshes in real time. When communication becomes available, robots engage in *local communication* to detect loop closures and perform distributed trajectory estimation. From the globally optimized trajectory estimates, each robot performs local mesh optimization to refine its local map. We also presented D-GNC, a novel two-stage method for

robust distributed pose graph optimization, which serves as the estimation backbone of Kimera-Multi and outperforms prior outlier rejection methods. Based on offline experiments using photo-realistic simulations and real-world datasets, we validated that Kimera-Multi (i) provides robust and accurate trajectory estimation while being fully distributed, (ii) estimates 3D meshes with improved metric-semantic accuracy compared to inputs from Kimera, and (iii) is communication-efficient and achieves significant communication reductions.

Furthermore, we conclude this chapter with results from large-scale field experiments of Kimera-Multi and a discussion about lessons learned. As a result of our field tests, we also contributed new challenging multi-robot CSLAM datasets with accurate reference trajectories and maps. Our datasets feature many robots traversing diverse indoor and outdoor scenes, facing additional challenges such as severe visual ambiguities and dynamic objects. We perform extensive quantitative analysis including comparisons of Kimera-Multi against a centralized baseline under simulated communication disruptions. These results further validate the accuracy and resilience of our system while identifying key factors affecting its performance.

Chapter 7

Collaborative Geometric Estimation with Event-Triggered Communication

7.1 Introduction

In this chapter and the next chapter, we shift our focus to tackle collaborative geometric estimation under the server-client communication architecture (Figure 1-1b). We first consider a generic problem setup where multiple agents (*e.g.*, robots, smartphones) coordinate with a central server to estimate a shared geometric model (*e.g.*, a 3D map). Recall that despite the use of a server, the underlying computation is still *distributed* across all agents. While distributing the computation promises better scalability and mitigates privacy concerns, it also requires more frequent communication due to the iterative nature of most optimization algorithms. Furthermore, the amount of data communicated at each iteration often grows proportionally with the dimension of the shared model. For large models, this type of *iterative communication* can result in long delays under real-world communication networks.

In this chapter, we address the aforementioned technical challenge by developing a *communication-efficient* algorithm for collaborative geometric estimation, which significantly reduces the burden of communication when performing distributed optimization on high-dimensional problems. The core idea behind our approach is the so-called *lazy* or *event-triggered* communication: instead of uploading all information

at every iteration, agents selectively upload parts of their local information that have changed significantly from the past. While the main idea is intuitive, incorporating lazy communication in geometric estimation applications raises a series of technical questions ranging from algorithm design to theoretical analysis of convergence that we address in this chapter.

Contributions. This chapter presents a *communication-efficient* distributed Riemannian optimization algorithm for collaborative geometric estimation. To tackle the numerical poor conditioning associated with most real-world problems, we design a distributed method that performs *approximate second-order* updates while simultaneously protecting the privacy of participating agents. Furthermore, we augment our basic method with *lazy communication*, which enables agents to only transmit the parts of their local information that satisfy certain *communication triggering conditions*, and hence significantly reduces overall communication. We prove that our final algorithm converges globally to first-order critical points with a global sublinear rate. Compared to related works that study lazy communication in distributed first-order methods (*e.g.*, [164]), our algorithm design and convergence analysis are significantly different and account for the employed second-order updates, the treatment of non-convex manifold constraints, among other details (see Remark 7.1). We perform extensive evaluations on large-scale BA problems in collaborative SLAM and SfM scenarios, which are central to emerging multi-robot navigation and mixed reality applications. Results show that our algorithm achieves competitive performance compared to other state-of-the-art methods under the same communication architecture, while achieving up to 78% total communication reduction.

Notations

We introduce several additional notations that are necessary for developing the algorithms in this chapter. We use calligraphic letters $\mathcal{X}, \mathcal{Y}$ to denote smooth matrix manifolds. Let $M : T_{x_1}\mathcal{X} \to T_{x_2}\mathcal{X}$ be a linear map between two tangent spaces of a smooth matrix manifold $\mathcal{X}$. With a slight abuse of notation, we also use M to denote the matrix representation of this linear map under chosen bases of $T_{x_1}\mathcal{X}$ and $T_{x_2}\mathcal{X}$.

For a tangent vector $u \in T_{x_1}\mathcal{X}$, $Mu \in T_{x_2}\mathcal{X}$ denotes the result of applying M on u. Further, $\|M\|$ denotes the operator norm of M with respect to the Riemannian metric. When $M : T_x\mathcal{X} \to T_x\mathcal{X}$ maps a tangent space to itself and is symmetric and positive definite, we define its associated inner product as $\langle u_1, u_2 \rangle_M \triangleq \langle u_1, Mu_2 \rangle$ with the corresponding norm $\|u\|_M \triangleq \sqrt{\langle u, u \rangle_M}$.

7.2 Problem Formulation

We consider a generic collaborative geometric estimation scenario, where N agents navigate in a common environment, and seek to collaboratively estimate a shared geometric model $y \in \mathcal{Y}$. For this purpose, agents communicate with a central server, who is responsible for coordinating updates across the team. In practice, the server can be a lead agent or a base station. Motivated by most real-world applications, we assume that the shared model y consists of m smaller elements $y = \{y_1, \ldots, y_m\}$, where each element $y_l \in \mathcal{Y}_l$ corresponds to a single geometric primitive. For instance, when y represents a point cloud map, each y_l corresponds to a single 3D point. During navigation, each agent $i \in [N] \triangleq \{1, \ldots, N\}$ observes a subset of the shared model. In addition, agent i also maintains *private* variable $x_i \in \mathcal{X}_i$, which can contain sensitive information such as the trajectory of this agent.

We focus on solving the maximum likelihood estimation (MLE) problem in the multi-agent scenario described above. Under the MLE formulation, local measurements collected by agent i induce a local cost function[1] $f_i : \mathcal{X}_i \times \mathcal{Y} \to \mathbb{R}$ that is usually *non-convex*. Under the standard assumption that agents' measurements are corrupted by independent noise, the global MLE problem takes the following form.

[1]For clarity of presentation, we assume that f_i depends on the entire shared model y. See Remark 7.2 for discussions of the general case.

Problem 7.1 (Generic MLE Formulation for Collaborative Geometric Estimation)**.**

$$\min_{x_i, y} \quad f(x, y) \triangleq \sum_{i=1}^{N} f_i(x_i, y), \tag{7.1a}$$

$$\text{s.t.} \quad x_i \in \mathcal{X}_i, \ \forall i \in [N], \ \ y \in \mathcal{Y}. \tag{7.1b}$$

In (7.1), we use $x \in \mathcal{X}$ to denote the concatenation of all private variables x_i, $i \in [N]$. The MLE formulation is very general and encompasses a wide range of robot perception problems [19]. In particular, Problem 7.1 includes collaborative bundle adjustment (Section 2.2.3) as a special case, as shown in the following example.

Example 1 (Collaborative Bundle Adjustment)**.** Bundle adjustment (BA) [42] is a crucial building block of modern visual SLAM and SfM systems. Recall from Section 2.2.3 that in collaborative BA, agents jointly estimate a global map using local measurements collected by monocular cameras. Assuming known camera intrinsics, the private variable x_i contains camera poses of agent i, *i.e.*, $x_i = \{T_1^{(i)}, \ldots, T_{n_i}^{(i)}\} \in \mathrm{SE}(3)^{n_i}$. The shared variable y consists of points in the global map, *i.e.*, $y = \{y_1, y_2, \ldots, y_m\} \in \mathbb{R}^{3\times m}$. Under the standard Gaussian noise model, the local cost function is given by the sum of local squared reprojection errors,

$$f_i(x_i, y) = \sum_{j=1}^{n_i} \sum_{l \in L_{ij}} w_{jl}^{(i)} \left\| q_{jl}^{(i)} - \pi(T_j^{(i)}, y_l) \right\|_2^2. \tag{7.2}$$

In (7.2), $L_{ij} \subseteq [m]$ denotes the set of points observed by agent i at pose $T_j^{(i)}$, $\pi(\cdot, \cdot)$ is the camera projection model, $q_{jl}^{(i)} \in \mathbb{R}^2$ denotes noisy observation on the image plane, and $w_{jl}^{(i)} > 0$ is the corresponding measurement weight. Note that (7.2) is essentially the same as the problem formulation in Section 2.2.3. The only difference is that we introduce an extra index $i \in [N]$ to distinguish between multiple agents.

In addition to collaborative BA, Problem 7.1 also includes other important types of problems. In what follows, we give two additional examples.

Example 2 (Collaborative Point Cloud Registration)**.** Multiple point cloud registration (*e.g.*, [201]) is an important problem with robotic applications such as merging multiple point cloud maps or collaborative SLAM with range sensors. In this case, the private and shared variables are the same as Example 1. The local cost function is given by,

$$f_i(x_i, y) = \sum_{j=1}^{n_i} \sum_{l \in L_{ij}} w_{jl}^{(i)} \left\| y_l - R_j^{(i)} q_{jl}^{(i)} - t_j^{(i)} \right\|_2^2, \tag{7.3}$$

where $T_j^{(i)} = (R_j^{(i)}, t_j^{(i)}) \in \mathrm{SE}(3)$ denote the rotation matrix and translation vector of the jth pose of agent i, and $q_{jl}^{(i)} \in \mathbb{R}^3$ denotes noisy 3D observation in the local frame.

Example 3 (Collaborative Object-Based PGO)**.** In some applications, it suffices to produce an object-level map of the environment (*e.g.*, [38]). In this case, the set of shared variables becomes $y = \{T_1, T_2, \ldots, T_m\} \in \mathrm{SE}(3)^m$, where T_l is the pose of object l, and the optimization problem becomes an instance of pose graph optimization (PGO). The local cost function (using chordal distance) is given by,

$$f_i(x_i, y) = \sum_{j=1}^{n_i} \sum_{l \in L_{ij}} w_{jl}^{(i)} \left\| T_l - T_j^{(i)} \, \widetilde{T}_{jl}^{(i)} \right\|_{\Omega_{jl}^{(i)}}^2, \tag{7.4}$$

where $\widetilde{T}_{jl}^{(i)} \in \mathrm{SE}(3)$ is a noisy relative measurement of object l collected by agent i at pose $T_j^{(i)}$, and $\Omega_{jl}^{(i)}$ is the corresponding measurement precision matrix.

In this chapter, we focus on collaborative BA (Example 1) in our experimental validation (Section 7.5), due to its fundamental role in multi-robot visual SLAM [1, 2, 4, 43]. However, we note that our approach extends beyond the above examples to many other multi-agent estimation problems that can be described with a factor graph [19]; see Remark 2.1 for additional discussions.

7.3 Proposed Algorithm

In this section, we present our communication-efficient distributed algorithm for solving Problem 7.1. In Section 7.3.1, we develop the basic form of our method based on

distributed approximate second-order updates. Similar to DDF-SAM [55, 57], in each iteration our method analytically eliminates the updates to private variables, which leads to more effective updates and also protects the privacy of participating agents. However, unlike DDF-SAM, our method avoids the transmission of dense matrices resulting from elimination, which makes it applicable to larger scale problems. Furthermore, in Section 7.3.2, we augment our basic method with *lazy communication*, which achieves significant communication reduction. Lastly, Section 7.3.3 summarizes the discussion and presents the complete algorithm.

7.3.1 Distributed Update with Analytic Elimination

At each iteration, agents collaboratively compute an updated solution that decreases the global cost in Problem 7.1. To start, each agent i constructs a second-order approximation $\widehat{m}_i$ for its local cost f_i, which is defined at the tangent space of the current iterate (x_i, y) at iteration k. For now, we drop the iteration index k to ease the burden of notations.

Intuitively, $\widehat{m}_i$ approximates the true local cost f_i when perturbing x_i and y on the tangent space. Formally, given tangent vectors $(u_i, v) \in T_{x_i}\mathcal{X}_i \times T_y\mathcal{Y}$, we define[2]

$$\widehat{m}_i(u_i, v) \triangleq f_i(x_i, y) + \left\langle \underbrace{\begin{bmatrix} g_{ix} \\ g_{iy} \end{bmatrix}}_{g_i}, \begin{bmatrix} u_i \\ v \end{bmatrix} \right\rangle + \frac{1}{2} \left\langle \begin{bmatrix} u_i \\ v \end{bmatrix}, \underbrace{\begin{bmatrix} A_i & C_i \\ C_i^\top & B_i \end{bmatrix}}_{M_i} \begin{bmatrix} u_i \\ v \end{bmatrix} \right\rangle. \tag{7.5}$$

In (7.5), $g_i \triangleq \operatorname{grad} f_i(x_i, y)$ is the local Riemannian gradient. The user-specified linear map $M_i \succ 0$ serves as an approximation of the local Riemannian Hessian, and is assumed to be symmetric and positive definite. For geometric estimation problems such as BA (7.2), we obtain the second-order approximation via the Riemannian Levenberg–Marquardt (LM) method [17, Chapter 8]. In this case, we have $M_i = J_i^\top J_i + \lambda I$, where J_i is the Jacobian of agent i's measurement residuals, and $\lambda > 0$ is a regularization parameter that ensures M_i to be positive definite.

[2]Note that $\widehat{m}_i$ depends on the linearization points x_i and y. We drop this from our notation for simplicity.

Given the local approximations $\widehat{m}_i$, a second-order approximation of the global cost f is given by $\widehat{m}(u,v) \triangleq \sum_{i=1}^{N} \widehat{m}_i(u_i, v)$, where we use u to denote the concatenation of local tangent vectors u_i. Note that $\widehat{m}$ can be expanded as,

$$\widehat{m}(u,v) = f(x,y) + \Big\langle \underbrace{\begin{bmatrix} g_x \\ g_y \end{bmatrix}}_{g}, \begin{bmatrix} u \\ v \end{bmatrix} \Big\rangle + \frac{1}{2} \Big\langle \begin{bmatrix} u \\ v \end{bmatrix}, \underbrace{\begin{bmatrix} A & C \\ C^\top & B \end{bmatrix}}_{M} \begin{bmatrix} u \\ v \end{bmatrix} \Big\rangle. \tag{7.6}$$

It can be verified that $g = \operatorname{grad} f(x,y)$ is the Riemannian gradient of the global objective. The linear map M in (7.6) is now an approximation of the global Riemannian Hessian. More importantly, M is a block matrix with an *arrowhead sparsity pattern*, and its blocks are related to the blocks of M_i in (7.5) as follows,

$$A = \operatorname{Diag}(A_1, \ldots, A_N), \;\; B = \sum_{i=1}^{N} B_i, \;\; C^\top = \begin{bmatrix} C_1^\top \ldots C_N^\top \end{bmatrix}. \tag{7.7}$$

In the proposed method, we seek to compute an update for all variables by approximately minimizing $\widehat{m}$. To proceed, we *analytically eliminate* private vector u from (7.6). Formally, define $u^\star(v) \triangleq \arg\min_u \widehat{m}(u,v)$ as the optimal private vector conditioned on the shared vector. Furthermore, define the *reduced second-order approximation* as $\widehat{h}(v) \triangleq \widehat{m}(u^\star(v), v)$, which only involves the shared vector v. Both $u^\star(v)$ and $\widehat{h}(v)$ admit closed-form expressions.

Lemma 7.1 (Reduced second-order approximation)**.** *For each agent $i \in [N]$, the corresponding optimal private vector is,*

$$u_i^\star(v) = -A_i^{-1}(C_i v + g_{ix}), \;\; \forall i \in [N]. \tag{7.8}$$

Furthermore, $\widehat{h}(v)$ has the closed-form expression,

$$\widehat{h}(v) = f(x,y) - \frac{1}{2}\left\langle g_x, A^{-1} g_x \right\rangle + \langle w, v \rangle + \frac{1}{2}\langle v, Sv \rangle, \tag{7.9}$$

where vector w and matrix S are defined as,

$$w \triangleq \sum_{i=1}^{N} w_i, \quad w_i \triangleq g_{iy} - C_i^\top A_i^{-1} g_{ix}, \ \forall i \in [N]. \tag{7.10}$$

$$S \triangleq \sum_{i=1}^{N} S_i, \quad S_i \triangleq B_i - C_i^\top A_i^{-1} C_i, \ \forall i \in [N]. \tag{7.11}$$

In the following, we refer to w in (7.10) and S in (7.11) as the *reduced gradient* and *reduced Hessian*, respectively. The analytic elimination technique presented above has been widely used to solve SLAM and BA [42], and is a special case of the *variable projection* approach to solve nonlinear least squares problem [202]. In the distributed setting, Lemma 7.1 suggests that the server can first aggregate w_i and S_i from all agents, and then minimize $\widehat{h}(v)$ by computing $v^\star = -S^{-1}w$. This type of approach has been proposed by DDF-SAM [55, 57]. Nevertheless, for large-scale problems such as BA, this approach is less suitable as it requires the communication of the S_i matrices, which are generally *dense* and thus expensive to evaluate, store, and transmit.

To design a communication-efficient update, we instead resort to finding an approximate minimizer of $\widehat{h}(v)$. In the following, let k denote the iteration number, and let w^k and S^k denote the values of the reduced gradient and reduced Hessian at iteration k. We let our approximate minimizer of $\widehat{h}(v)$ take the following form,

$$v^k \triangleq -\gamma P^k w^k, \tag{7.12}$$

where $\gamma > 0$ is a constant stepsize, and P^k is a *sparse* matrix that approximates the inverse of the reduced Hessian S^k. Viewing w^k as the reduced gradient and P^k as a preconditioner, we may interpret (7.12) as a single step of preconditioned Riemannian gradient descent. In the following, we use the block Jacobi preconditioner [176] due to its simplicity,

$$P^k = \left(\sum_{i=1}^{N} D_i^k \right)^{-1}, \quad D_i^k \triangleq \mathrm{Diag}(S_{i,1}^k, \ldots, S_{i,m}^k), \tag{7.13}$$

where $S_{i,l}^k$ is the l-th diagonal block of S_i^k. Note that each block l corresponds to a single element y_l in the shared variable y (see Section 7.2). With (7.12) and (7.13), agent i only needs to upload w_i^k and the diagonal blocks of S_i^k to the server. Furthermore, the server can easily compute P^k as it only requires inverting a block-diagonal matrix.

Once the shared update v^k is computed, we leverage Lemma 7.1 to compute the corresponding optimal second-order update for each agent's private variable,

$$u_i^k \triangleq u_i^\star(v^k),\ \forall i \in [N]. \tag{7.14}$$

Finally, we compute the updated estimates using retraction,

$$y^{k+1} = \mathrm{Retr}_{y^k}(v^k),\ x_i^{k+1} = \mathrm{Retr}_{x_i^k}(u_i^k), \forall i \in [N], \tag{7.15}$$

and the algorithm proceeds to the next iteration.

7.3.2 Incorporating Lazy Communication

In the method developed so far, at each iteration, agent i needs to upload w_i and D_i to the server. For larger problems, the resulting transmission can still become too expensive. In this subsection, we present a technique to further reduce communication. The core idea behind our approach is *lazy communication*: when some blocks of w_i and D_i do not change significantly from previous iterations, agent i simply skips the transmission of those blocks, and the server reuses values received at previous iterations for its computation. In the following, we describe this process in detail for the computation of preconditioner and the reduced gradient, respectively. Without loss of generality, we present our method from the perspective of agent i.

Lazy communication of preconditioner. Let k be the current iteration number. For each block l, let $k' < k$ be the last iteration when agent i uploads $S_{i,l}^{k'}$ to the

server.[3] Using $S_{i,l}^{k'}$, we can compute an approximation of $S_{i,l}^{k}$ as,

$$\widetilde{S}_{i,l}^{k} \triangleq T_{l}^{k \leftarrow k'} \circ S_{i,l}^{k'} \circ T_{l}^{k' \leftarrow k}, \tag{7.16}$$

where $T_{l}^{k' \leftarrow k}$ is the matrix that represents a *transporter* [18, Sec. 10.5] from the tangent space at iteration k to iteration k', and $T_{l}^{k \leftarrow k'}$ is its adjoint. Intuitively, transporters are needed to ensure that the approximation defined in (7.16) represents a valid linear map on the tangent space at iteration k. For matrix manifolds, a simple and computationally efficient transporter can be obtained from orthogonal projections to tangent spaces [18, Proposition 10.60]. Note that since (7.16) only uses past information, both the server and agent i can compute $\widetilde{S}_{i,l}^{k}$ *without any communication.*

The above approximation leads to the following lazy communication scheme. First, agent i compares $S_{i,l}^{k}$ and its approximate version $\widetilde{S}_{i,l}^{k}$ locally. Then, agent i only uploads $S_{i,l}^{k}$ to the server if the approximation error is large,

$$\left\| S_{i,l}^{k} - \widetilde{S}_{i,l}^{k} \right\| > \delta_p \left\| S_{i,l}^{k} \right\|, \tag{7.17}$$

where $\delta_p \geq 0$ is a user defined threshold. On the other hand, if (7.17) does not hold (*i.e.*, approximation error is small), agent i skips the communication of $S_{i,l}^{k}$, and the server uses the approximation $\widetilde{S}_{i,l}^{k}$ instead.

In summary, for each agent i, instead of using D_i^k as defined in (7.13), the server now uses an approximation $\widehat{D}_i^k$ that consists of a mixture of exact and approximate blocks. Specifically, the l-th diagonal block of $\widehat{D}_i^k$ is given by,

$$\widehat{D}_{i,l}^{k} \triangleq \begin{cases} S_{i,l}^{k}, & \text{if (7.17) holds,} \\ \widetilde{S}_{i,l}^{k}, & \text{otherwise.} \end{cases} \tag{7.18}$$

Finally, the preconditioner becomes $P^k = (\sum_{i=1}^{N} \widehat{D}_i^k)^{-1}$.

Lazy communication of reduced gradient. We can employ a similar strategy

[3] For notation simplicity, we drop the dependence of k' on i and l.

to design a lazy communication rule for the transmission of the reduced gradient w_i, which is needed by the server to compute the update step in (7.12). For this purpose, let us view w_i as a block vector, where each block $w_{i,l}$ corresponds to a single element y_l in the shared variable. For each block l, let $k' < k$ be the last iteration when agent i uploads $w_{i,l}^{k'}$ to the server. Using $w_{i,l}^{k'}$, we can compute an approximation of $w_{i,l}^{k}$ as follows,

$$\widetilde{w}_{i,l}^{k} \triangleq T_l^{k \leftarrow k'} \left(w_{i,l}^{k'} \right). \tag{7.19}$$

Once again, a transporter is needed to ensure (7.19) defines a valid tangent vector on the tangent space at the current iteration k. Similar to (7.16), computing (7.19) does not require communication between the server and agent i.

Similar to the previous development, at each iteration, agent i only uploads $w_{i,l}^{k}$ to the server if it differs significantly from $\widetilde{w}_{i,l}^{k}$. Specifically, we define the following communication triggering condition,

$$\left\| \widetilde{w}_{i,l}^{k} - w_{i,l}^{k} \right\|_{P_l^k}^2 > \frac{1}{mN^2} \sum_{d=1}^{\bar{d}} \epsilon_d \left\| \widehat{w}^{k-d} \right\|_{P^{k-d}}^2 . \tag{7.20}$$

The left-hand side of (7.20) measures the approximation error, using the norm associated with the current preconditioner of this block. The right-hand side defines a threshold on the approximation error using information from the past $\bar{d}$ iterations. Specifically, $\widehat{w}^{k-d}$ is the approximate reduced gradient used by the server at iteration $k-d$, and its exact definition is provided in (7.22). Both $\bar{d}$ and weights $\{\epsilon_d \geq 0, d = 1, \ldots \bar{d}\}$ are user specified constants. Note that setting $\epsilon_d = 0$ forces the agent to always upload.

Consequently, on the server's side, instead of using the up-to-date w_i^k vector for agent i, it uses an approximate version $\widehat{w}_i^k$ that consists of both exact and approximate blocks,

$$\widehat{w}_{i,l}^{k} \triangleq \begin{cases} w_{i,l}^{k}, & \text{if (7.20) holds,} \\ \widetilde{w}_{i,l}^{k}, & \text{otherwise.} \end{cases} \tag{7.21}$$

Finally, instead of using w^k to compute the update in (7.12), the server uses the

Algorithm 7.1 LARPG

Initialize solution x^0, y^0.
for iteration $k = 0, 1, \ldots$ **do**
  In parallel, agent i computes the local second-order approximation $\widehat{m}_i$ (7.5), and evaluates w_i^k (7.10) and D_i^k (7.13)
  // Lazy communication of preconditioner
  for each agent i **in parallel do**
    for each block l **do**
      Upload $S_{i,l}^k$ to server if (7.17) is true
    end for
  end for
  Server collects uploads and forms $\widehat{D}_i^k$ (7.18) for each agent
  Server computes preconditioner $P^k = (\sum_{i=1}^N \widehat{D}_i^k)^{-1}$ and broadcasts to agents
  // Lazy communication of reduced gradient
  for each agent i **in parallel do**
    for each block l **do**
      Upload $w_{i,l}^k$ to server if (7.20) is true
    end for
  end for
  Server collects uploads and forms $\widehat{w}_i^k$ and $\widehat{w}^k$
  // Compute update vector and next iterate
  Server computes $v^k = -\gamma P^k \widehat{w}^k$ and broadcasts to agents
  In parallel, each agent computes $u_i^k = u_i^\star(v^k)$; *c.f.* (7.8)
  Both server and agents update iterates

$$y^{k+1} = \mathrm{Retr}_y(v^k),\ x_i^{k+1} = \mathrm{Retr}_{x_i^k}(u_i^k).$$

end for

approximation defined as,

$$\widehat{w}^k \triangleq \sum_{i=1}^N \widehat{w}_i^k. \tag{7.22}$$

We conclude this subsection by noting that the lazy communication condition for reduced gradient (7.20) is more complex than the condition for preconditioner (7.17). The more complex rule (7.20) is needed for our convergence analysis, and we provide more discussions in Section 7.4.

7.3.3 The Complete Algorithm

We collect the steps discussed above and present the Lazily Aggregated Reduced Preconditioned Gradient (LARPG) algorithm with the complete pseudocode in Algorithm 7.1. Each iteration of LARPG has three stages. The first stage (lines 4-11) performs the lazy communication of the preconditioner. The second stage (lines 12-18) performs the lazy communication of the reduced gradients. We note that this stage needs to happen after the first stage, as the triggering rule for the reduced gradient (7.20) depends on the preconditioner P^k. The third stage (lines 19-22) uses the lazily aggregated information to compute the next iterate of the algorithm.

Remark 7.1 (Novelty with respect to [164])**.** Our lazy communication scheme is inspired by Chen *et al.* [164], who study lazily aggregated gradient methods in distributed optimization. However, our algorithm and analysis (Section 7.4) consists of the following important innovations to account for the unique challenges of Problem 7.1: (i) we consider problems with *non-convex* manifold constraints that are prevalent in robot perception applications, (ii) we incorporate the use of approximate second-order updates that require substantial changes in the convergence analysis, (iii) we handle private variables via analytic elimination, and (iv) we propose lazy communication on individual blocks of the gradient and preconditioner, which leads to further communication reduction.

Remark 7.2 (Implementation)**.** In many applications, such as collaborative SLAM, each agent i only observes parts of the shared model during navigation. Consequently, the local cost f_i only depends on the observed subset of the shared variable y. In our implementation and experiments (Section 7.5), we account for this fact by performing lazy communication *only* on the observed parts of y for each agent.

7.4 Convergence Analysis

Since LARPG allows agents to lazily upload information to the server, it is unclear if the algorithm can converge to a desired solution in general. In this section, we

provide a rigorous answer to this important question. In particular, we show that under several technical conditions, LARPG provably converges to a first-order critical point of Problem 7.1, despite the use of lazy communication. Assumption 7.1 below summarizes these technical assumptions.

Assumption 7.1. There exist constants $L > \mu \geq c_g > 0$ and $\mu_p, \sigma_p > 0$ such that the following conditions hold at any iteration $k \in \mathbb{N}$ of Algorithm 7.1,

A1 (Lipschitz-type gradient for pullbacks [167]) Let $f^k \triangleq f(x^k, y^k)$ and $g^k \triangleq \operatorname{grad} f(x^k, y^k)$ denote the objective and Riemannian gradient at iteration k. The *pullback function* $\widehat{f^k}(u, v) \triangleq f(\operatorname{Retr}_{x^k}(u), \operatorname{Retr}_{y^k}(v))$ satisfies

$$\left| \widehat{f^k}(u,v) - \left[f^k + \left\langle \begin{bmatrix} g_x^k \\ g_y^k \end{bmatrix}, \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle \right] \right| \leq \frac{c_g}{2} \left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\|^2, \tag{7.23}$$

for all $(u, v) \in T_{x^k}\mathcal{X} \times T_{y^k}\mathcal{Y}$.

A2 (Bounded Hessian approximation) At any iteration k, the approximate Hessian M^k defined in (7.6) satisfies $\mu I \preceq M^k \preceq LI$.

A3 (Preconditioner) At any iteration k, the preconditioner P^k used in Algorithm 7.1 (line 11) satisfies $P^k \succeq \mu_p I$ and $\left\| S^k P^k \right\|_{P^k} \leq \sigma_p$.

Above, (A1) is first introduced in [167] as a generalization of the standard Lipschitz smoothness assumption to the Riemannian setting. Intuitively, (A1) bounds the pullback function by its local linearization. Prior work in distributed BA [71, 72] requires similar smoothness conditions for convergence. In comparison, our assumptions and convergence guarantees extend beyond BA and hold for more general problems. (A2) assumes that the employed Hessian approximation M is bounded, which is also a standard assumption. Lastly, (A3) assumes that the preconditioner P^k is sufficiently positive definite, and the approximation error of P^k as the inverse of the reduced Hessian S^k is bounded.

The key to our convergence analysis (inspired by [164]) is to study the iterates of LARPG with respect to a *Lyapunov function*,

$$V^k \triangleq f(x^k, y^k) + \sum_{d=1}^{\bar{d}} \beta_d \left\| \widehat{w}^{k-d} \right\|^2_{P^{k-d}}. \tag{7.24}$$

At iteration k, V^k combines the cost function with weighted squared norms of past approximate reduced gradients. Intuitively, these squared norms account for the approximation errors caused by lazy communication. In Appendix C.2, we show that LARPG is a *descent* method with respect to V^k. This enables us to establish our main theoretical result.

Theorem 7.1. *Under Assumption 7.1, there exist suitable choices of algorithm parameters $\gamma, \bar{d}$, and $\{\epsilon_d \geq 0, d = 1, \ldots, \bar{d}\}$ such that after K iterations, the iterates generated by Algorithm 7.1 satisfy,*

$$\min_{k \in [K]} \left\| \operatorname{grad} f(x^k, y^k) \right\|^2 = O(1/K). \tag{7.25}$$

In Appendix C.2, we prove Theorem 7.1 and provide explicit parameter settings that guarantee convergence. The established $O(1/K)$ convergence rate matches standard global convergence result in Riemannian optimization [167]. While our convergence conditions involve additional parameters, experiments (Section 7.5) show that LARPG is not sensitive to these parameters and converges under a wide range of values.

7.5 Experimental Results

In this section, we evaluate LARPG on BA problems from benchmark collaborative SLAM and SfM datasets. All algorithms are implemented in C++ using g2o [8], and experiments are conducted on a computer with Intel i7-7700K CPU and 16 GB RAM. Unless otherwise mentioned, the default parameters we use for LARPG are

Table 7.1: Default parameters of LARPG used in experiments (Section 7.5).

Parameter	Value	Description
γ	1.0	Stepsize used to update shared variable (7.12).
λ	10^6	Regularization parameter in LM.
ϵ_d	10	Parameter used to compute the lazy communication threshold (7.20).
$\bar{d}$	10	Number of past gradients considered when computing lazy communication threshold (7.20).
δ_p	0.1	Parameter used for updating the lazy Jacobi preconditioner (7.17).

summarized in Table 7.1. Our results show that LARPG converges under a wide range of parameter settings, and compares favorably against existing methods while achieving up to 78% total communication reduction. In the rest of this section, we first perform ablation studies on the proposed lazy communication scheme (Section 7.5.1). Then, we present evaluation and comparison results on large-scale benchmark datasets (Section 7.5.2 and 7.5.3).

7.5.1 Evaluating Lazy Communication

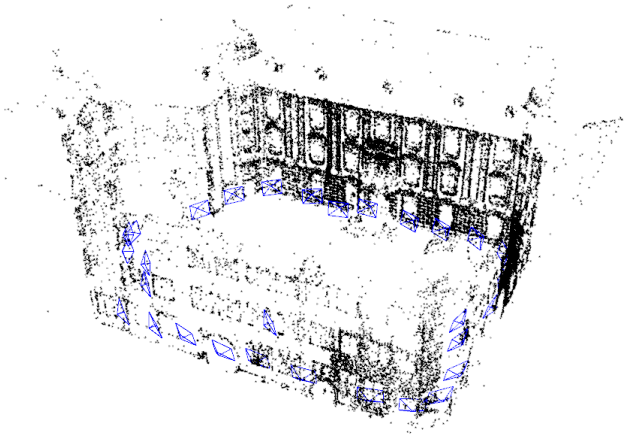
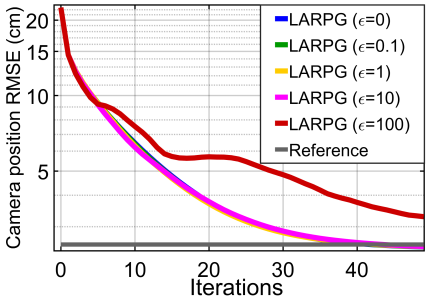
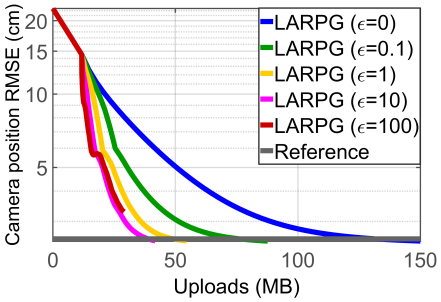


Figure 7-1: Castle30 dataset.

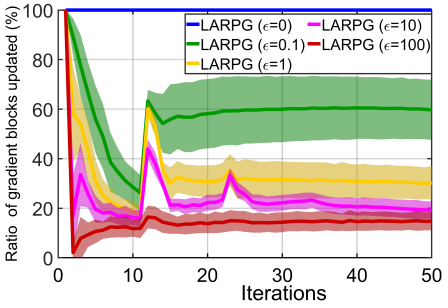
We evaluate the proposed lazy communication scheme using the Castle30 dataset [203], which consists of 30 images observing a courtyard (Figure 7-1). We use Theia [204] to generate the input BA problem, which contains 23564 map points in total. We divide the BA problem into 30 agents and run LARPG for 50 iterations. In this



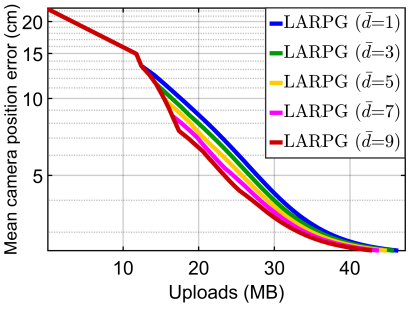
(a) Effect of varying ϵ (RMSE vs. iteration)



(b) Effect of varying ϵ (RMSE vs. total uploads)



(c) Effect of varying ϵ (% gradient blocks uploaded)



(d) Effect of varying $\bar{d}$ (RMSE vs. total uploads)

Figure 7-2: Evaluation of lazy communication on Castle30 dataset [203]. We evaluate the performance of LARPG under varying values of parameters ϵ and $\bar{d}$ that control the behavior of lazy communication.

experiment, we find that it is sufficient to fix the preconditioner at the initial iteration, which corresponds to letting $\delta_p \to +\infty$ in (7.17). This is because for this relatively simple problem, the initial preconditioner already gives a good approximation of curvature information at all subsequent iterates. Consequently, we mainly focus on evaluating parameters that affect the lazy communication of gradients (7.20).

We first evaluate the impact of ϵ_d in (7.20). Intuitively, larger values of ϵ_d imply that agents are more tolerant of gradient approximation error, and hence communicate less at each iteration. We set all ϵ_d $(d = 1, \ldots, D)$ to a common value ϵ and vary ϵ in our experiments. To measure solution accuracy, we record the root-mean-square error (RMSE) of camera positions, computed after aligning with the ground truth via a similarity transformation. Figure 7-2a shows the convergence of LARPG under varying values of ϵ. For comparison, we also include a reference solution computed by centralized optimization using g2o. Except when using a very loose threshold of

Table 7.2: Evaluation on collaborative SLAM scenarios [195, 205]. Columns **N, #IM, #MP, #Obs** denote the total number of agents, images (keyframes), map points, and observations, respectively. **Init**: input to all algorithms. **Ref**: reference solution from centralized optimization using g2o [8]. **PCG**: baseline distributed preconditioned conjugate gradient method [66]. **DR**: baseline Douglas-Rachford splitting method [71]. **LARPG**: proposed method ($\epsilon = 1$). For each metric, the best-performing distributed method is highlighted in bold.

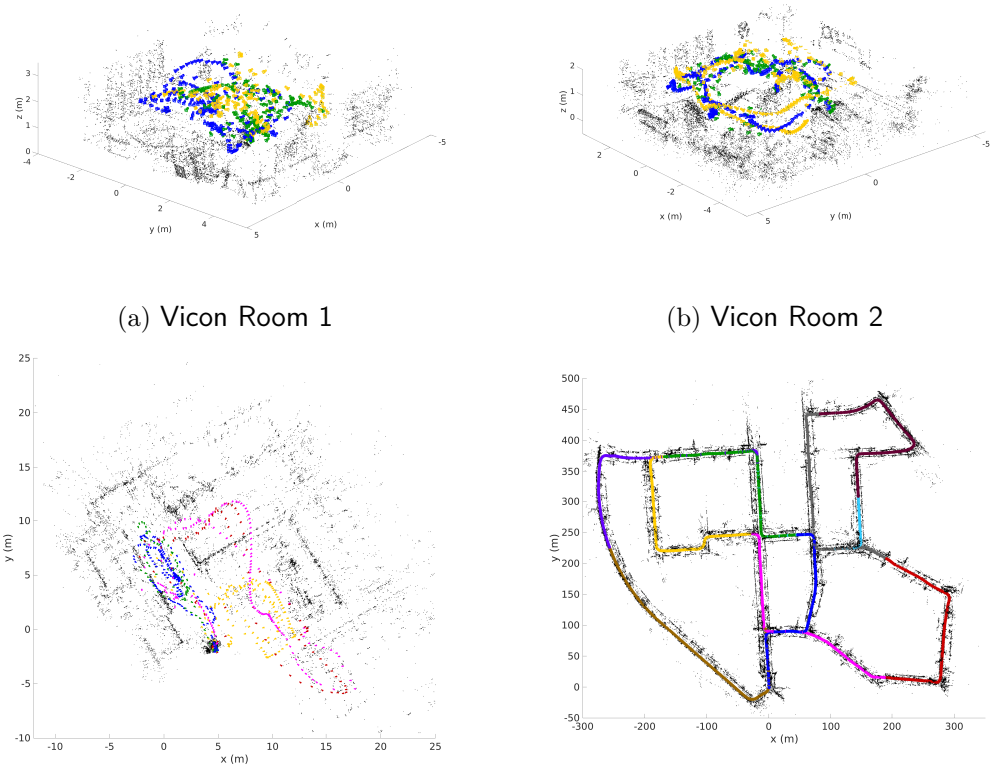
Dataset	N	#IM	#MP	#Obs	Absolute Trajectory Error (ATE) [m]					Mean Reprojection Error [px]					Total Uploads [MB]		
					Init	Ref	PCG	DR	**LARPG**	Init	Ref	PCG	DR	**LARPG**	PCG	DR	**LARPG**
Vicon Room 1	3	464	13K	121K	0.213	0.127	0.127	0.127	**0.126**	47.3	1.38	1.39	1.40	**1.38**	34	26	**11**
Vicon Room 2	3	631	20K	176K	0.191	0.087	0.089	**0.088**	**0.088**	45.3	1.42	1.51	1.46	**1.43**	43	32	**14**
Machine Hall	5	719	19K	187K	0.297	0.274	0.253	**0.215**	0.232	50.3	1.38	3.72	**1.38**	1.43	61	46	**17**
KITTI 00	10	1699	96K	553K	6.83	5.88	5.88	**5.86**	5.87	133.1	1.08	1.49	**1.09**	1.10	176	133	**71**
KITTI 06	10	422	22K	120K	10.87	10.32	10.42	**10.32**	10.36	107.9	1.11	**1.11**	1.12	**1.11**	44	34	**16**

$\epsilon = 100$ (red curve), lazy communication has minimal impact on the iterations of LARPG. Furthermore, the communication efficiency of our method is clearly seen in Figure 7-2b, where we plot convergence as a function of total amount of uploads to the server. To provide more insights, Figure 7-2c visualizes the amount of gradient blocks uploaded to the server at each iteration. For each value of ϵ, the corresponding solid line denotes the percentage of uploaded gradient blocks averaged across all agents, and the surrounding shaded area represents one standard deviation. Recall that choosing $\epsilon = 0$ forces all agents to upload all blocks at every iteration (blue curve in Figure 7-2c). Our result clearly shows that varying ϵ provides an effective way to control the amount of uploads during optimization.

In addition, we also evaluate the impact of $\bar{d}$ on convergence. Recall from (7.20) that $\bar{d}$ determines the number of past gradients that are used to compute the triggering threshold. Figure 7-2d shows the performance of LARPG under different choices of $\bar{d}$ with fixed $\epsilon = 5$. While the differences are not significant, our result still suggests that using more past gradients (*e.g.*, $\bar{d} = 9$) helps to save more communication.

7.5.2 Performance on Collaborative SLAM Datasets

In this subsection, we evaluate LARPG on collaborative BA problems from multi-robot SLAM applications. We use the monocular version of ORB-SLAM3 [37] to extract BA problems from the EuRoc [195] and KITTI [205] datasets. Figure 7-3 show



(a) Vicon Room 1

(b) Vicon Room 2

(c) Machine Hall

(d) KITTI 00

(e) KITTI 06

Figure 7-3: Visualization of BA problems in collaborative SLAM scenarios [195, 205], generated using ORB-SLAM3 [37]. Map points are shown in black. Poses of each simulated agent are shown in a distinct color.

visualizations of selected collaborative SLAM datasets. Each EuRoc dataset contains multiple sequences recorded in the same indoor space, and we use the multi-session feature of ORB-SLAM3 to simulate each sequence as a single robot. For each KITTI dataset, we divide the overall trajectory into multiple segments to simulate multiple robots. We generate noisy inputs for each dataset by perturbing the ORB-SLAM3

estimates by zero-mean Gaussian noise.[4]

We compare LARPG against two baseline methods that can be implemented under the communication architecture considered in this work. The first baseline is the method in [66] using distributed preconditioned conjugate gradient (PCG). In our case, we use distributed PCG to solve the reduced second-order approximation in Lemma 7.1, where the problem is re-linearized after every 10 PCG iterations. The second baseline is the Douglas-Rachford (DR) splitting method proposed in [71]. Similar to our method, both baseline methods only require agents to communicate information over the observed parts of the shared model (see Remark 7.2).

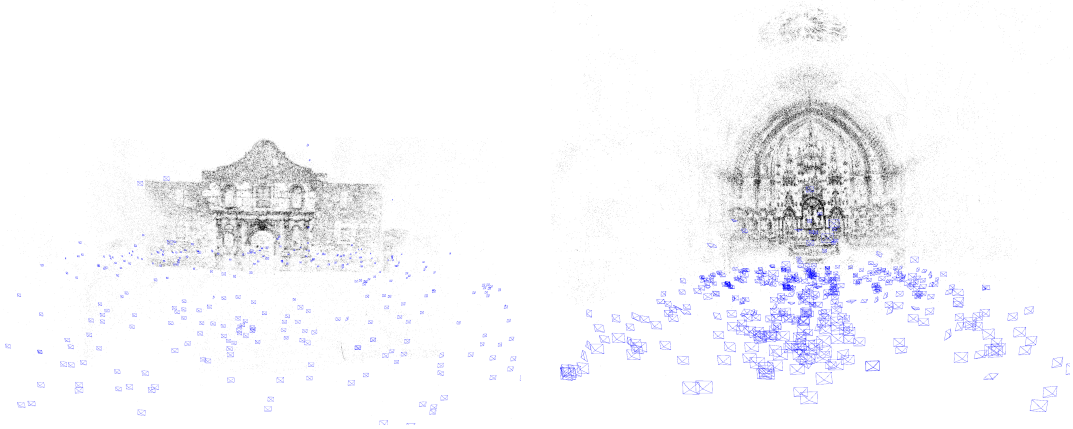
Table 7.2 shows the performance of all algorithms after 50 iterations. All results are averaged across 10 random runs. We evaluate the RMSE absolute trajectory error (ATE) against ground truth, the mean reprojection error, as well as the total amount of uploads during optimization. For comparison, we also include a reference solution computed by centralized optimization using g2o. We note that the higher ATE in KITTI is due to the larger scale of the datasets. As shown in Table 7.2, LARPG achieves similar or better performance compared to baseline methods, while using significantly less communication. Specifically, when compared to DR, LARPG achieves up to 65% total communication reduction, clearly demonstrating the communication efficiency of our method.

7.5.3 Performance on Collaborative SfM Datasets

We also evaluate LARPG on collaborative SfM scenarios using the 1DSfM dataset [206], which contains 15 medium to large scale internet photo collections. Figure 7-4 show visualizations of selected datasets. We use Theia [204] to generate the input BA problems. Then, we partition each problem randomly to simulate a scenario with 50 agents.

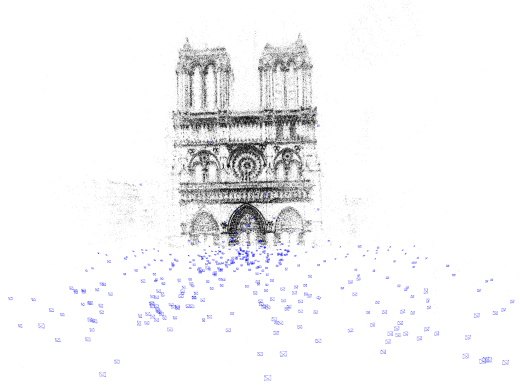
Similar to the previous subsection, we evaluate the performance of all algorithms after 50 iterations. Table 7.3 shows the results. Since ground truth is not available, we

[4]Specifically, the noise standard deviation for robot rotation, robot position, and map points are set to 5 deg, 0.1 m, 0.05 m for EuRoc, and 5 deg, 2 m, 0.1 m for KITTI.

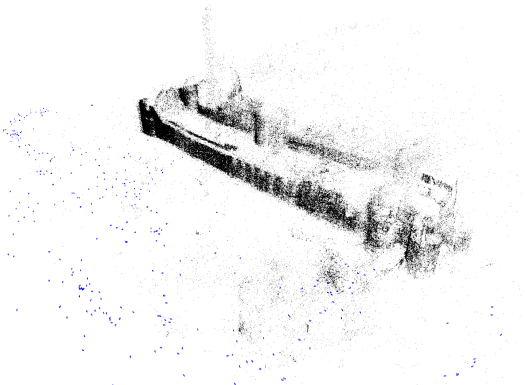


(a) Alamo

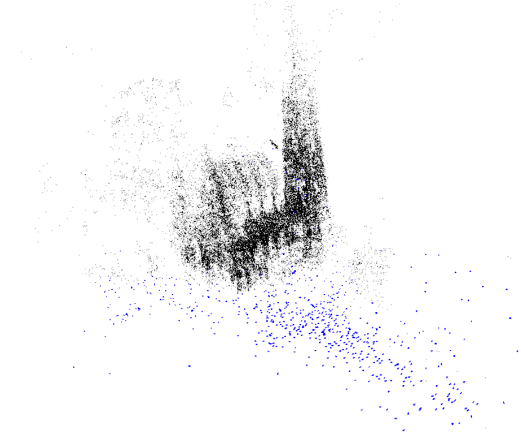
(b) Montreal Notre Dame



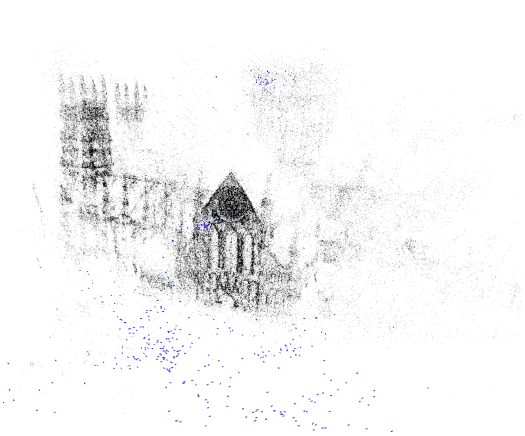
(c) Notre Dame



(d) Tower of London



(e) Vienna Cathedral



(f) Yorkminster

Figure 7-4: Visualization of BA problems in collaborative SfM scenarios [206], generated using Theia [204]. Map points and camera poses are shown in black and blue, respectively. Each dataset is partitioned into 50 agents to simulate a collaborative BA problem.

only record the final mean reprojection error. LARPG outperforms baseline methods in most datasets, and achieves final reprojection errors that are close to the central-

Table 7.3: Evaluation on collaborative SfM scenarios [206]. Each dataset is divided to simulate 50 agents. For LARPG, we set $\epsilon = 10$. All columns are named in the same way as Table 7.2. For each metric, the best performing distributed method is highlighted in bold.

Dataset	#IM	#MP	#Obs	Mean Reprojection Error [px]					Total Uploads [MB]			Average Local Iteration Time [ms]		
				Init	Ref	PCG	DR	**LARPG**	PCG	DR	**LARPG**	PCG	DR	**LARPG**
Alamo	576	138K	813K	2.56	1.39	1.57	1.63	**1.44**	989	745	**186**	**16**	151	76
Ellis Island	234	22K	86K	5.30	2.61	5.04	4.07	**3.24**	117	90	**22**	**1**	8	8
Gendarmenmarkt	704	78K	271K	4.34	2.02	2.96	2.67	**2.23**	379	286	**73**	**5**	35	27
Madrid Metropolis	345	45K	198K	3.77	1.28	**1.48**	1.87	1.49	272	205	**56**	**3**	23	19
Montreal Notre Dame	459	152K	811K	3.05	1.96	**2.04**	2.10	2.08	1048	790	**171**	**16**	124	80
Notre Dame	548	225K	1180K	3.97	2.18	2.34	2.87	**2.23**	1345	1014	**257**	**23**	239	109
NYC Library	336	54K	210K	3.67	1.72	2.17	2.21	**1.89**	294	222	**57**	**4**	24	21
Piazza del Popolo	336	31K	154K	4.63	1.88	2.54	2.33	**2.20**	199	150	**38**	**2**	14	14
Piccadilly	2303	185K	797K	4.64	2.11	3.72	3.27	**2.55**	972	733	**177**	**16**	159	80
Roman Forum	1067	227K	1046K	4.20	1.82	2.14	2.79	**1.90**	1400	1056	**279**	**21**	221	108
Tower of London	484	124K	557K	5.14	1.68	4.48	2.61	**2.50**	702	583	**127**	**12**	101	57
Trafalgar	5067	333K	1286K	4.80	2.11	3.76	3.24	**2.17**	1678	1265	**309**	**28**	293	146
Union Square	816	26K	90K	6.77	1.93	3.71	3.32	**2.91**	121	91	**22**	**1**	8	8
Vienna Cathedral	843	157K	504K	5.73	1.88	3.69	3.32	**2.37**	723	545	**146**	**11**	92	55
Yorkminster	428	101K	377K	5.29	2.02	2.99	3.16	**2.24**	542	409	**128**	**7**	59	39

ized reference solutions. Once again, LARPG demonstrates superior communication efficiency. When compared to DR, our method achieves 68%-78% reduction in terms of total uploads. Lastly, we also evaluate the average local iteration time of all methods (last three columns in Table 7.3). Our method is faster than DR, since the latter requires each agent to solve a smaller nonlinear optimization problem at every iteration. On the other hand, the local iteration time of our method is larger than PCG. However, considering the large size of the SfM datasets, an average iteration time ranging from 8 ms to 293 ms for our method is still reasonable, and can be improved by further optimizing our implementation (*e.g.*, via additional parallelization).

7.6 Conclusion

This chapter presented LARPG, a communication-efficient distributed algorithm for collaborative geometric estimation. Each iteration of LARPG allows agents to analytically eliminate private variables. Furthermore, by incorporating lazy and partial aggregation at the server, LARPG achieves significant communication reduction, which makes it suitable for multi-robot and mixed reality applications subject to lim-

ited network bandwidth. Under generic conditions, we proved that LARPG converges globally to first-order critical points with a sublinear convergence rate. Evaluations on large-scale BA problems in collaborative SLAM and SfM scenarios show that LARPG performs competitively against existing techniques while achieving a consistent communication reduction of up to 78%.

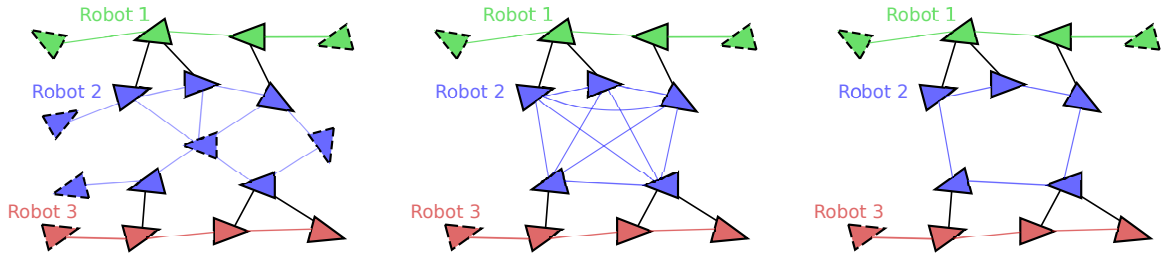
Chapter 8

Collaborative Rotation Averaging and Translation Estimation with Spectral Sparsification

8.1 Introduction

Chapter 7 develops a communication-efficient algorithm for generic collaborative geometric estimation under the server-client architecture. However, despite its generality, the LARPG algorithm only achieves sublinear convergence rate (Theorem 7.1); this means that on challenging problem instances, the algorithm could use many iterations to converge to high-precision solutions. In this chapter, we show that for the special problems of collaborative rotation averaging and translation estimation, we can in fact develop optimization algorithms that achieve much faster convergence while preserving communication efficiency. These problems are fundamental and have applications ranging from initialization for PGO [31], SfM [32], and camera network localization [33].

Similar to Chapter 7, the approach developed in this chapter is based on the server-client architecture (Figure 1-1b). The crux of our method lies in exploiting theoretical relations between the Hessians of the optimization problems and the



(a) Measurement graph (b) Robot 2's *dense* reduced graph (c) Robot 2's *sparsified* graph

Figure 8-1: (a) Example 3-robot problem visualized as a graph. For each robot $\alpha \in \{1, 2, 3\}$, its vertices (variables) $\mathcal{V}_\alpha$ are shown in a distinct color. Each edge corresponds to a relative measurement between two variables. Separators (solid line) correspond to variables with inter-robot measurements, and the remaining variables form the interior vertices (dashed line). (b) For robot 2, elimination of its interior vertices creates a *dense* matrix S_2, which corresponds to a dense graph over its separators. (c) In our approach, robot 2 achieves communication efficiency by transmitting a sparse approximation $\widetilde{S}_2$ of the original dense matrix S_2, which also corresponds to a sparsified graph over its separators.

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 approximately solving a Laplacian system of the form $Lx = b$. Importantly, before communication, robots use *spectral sparsification* [135, 136] to sparsify intermediate dense matrices resulted from elimination of its internal variables. Figure 8-1 shows 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. Furthermore, the theoretical properties of spectral sparsification allow us to perform rigorous convergence analysis, and establish *linear* rates of convergence for our methods. Lastly, we also present an extension to *outlier-robust* estimation by combining our approach with graduated non-convexity (GNC) [28, 189].

Contributions. The key contributions of this chapter are summarized as follows:

- We present collaborative optimization algorithms for multi-robot 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*.

- 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.
- We present an extension to *outlier-robust* estimation by combining the proposed algorithms with GNC.
- We perform extensive evaluations of our methods and demonstrate their values on real-world SLAM and SfM scenarios with outlier measurements.

Lastly, 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 collaborative pose graph optimization (PGO), which is another fundamental problem commonly used in collaborative SLAM (CSLAM).

Chapter Organization. The rest of this chapter is organized as follows. The remainder of this section introduces additional notations and mathematical preliminaries. Section 8.2 reviews the problem formulation. In Section 8.3, we establish theoretical relations between the Hessians and the underlying graph Laplacians. Then, in Section 8.4, we leverage these theoretical results to design fast and communication-efficient solvers for the problems of interest and establish convergence guarantees. Finally, Section 8.5 presents numerical evaluations of the proposed algorithms using large-scale datasets from CSLAM and SfM scenarios.

Notations and Preliminaries

In the following, we introduce additional notations and preliminaries that are necessary for developing the algorithms in this chapter. The reader is also referred to Table D.1 for a summary of detailed notations used in this chapter.

Spectral Approximation. Following [141, 142], for $A, B \in \mathcal{S}^n$ and $\epsilon > 0$, we say that B is an *ϵ-approximation* of A, denoted as $A \approx_\epsilon B$, if the following holds,

$$e^{-\epsilon} B \preceq A \preceq e^{\epsilon} B, \tag{8.1}$$

where $B \preceq A$ means $A - B \in \mathcal{S}_+^n$. Note that (8.1) is symmetric and holds under composition: if $A \approx_\epsilon B$ and $B \approx_\delta C$, then $A \approx_{\epsilon+\delta} C$. Furthermore, if A is singular, the relation (8.1) implies that B is necessarily singular and $\ker(A) = \ker(B)$.

Inner Products on the Tangent Spaces of $\mathrm{SO}(d)$. In this chapter, we define the inner products on the tangent space of $\mathrm{SO}(d)$ (where $d \in \{2, 3\}$) as follows. For any tangent vectors $\eta_1, \eta_2 \in T_R \mathrm{SO}(d)$, let $v_1, v_2 \in \mathbb{R}^p$ be their representation using the basis defined in Section 2.1 (eqs. (2.5) and (2.6)), *i.e.*,

$$\eta_1 = R\,[v_1]_\times\,, \quad \eta_2 = R\,[v_2]_\times\,. \tag{8.2}$$

We define the inner product as follows,

$$\langle \eta_1, \eta_2 \rangle_R \triangleq v_1^\top v_2. \tag{8.3}$$

It can be verified that the inner product defined above is related to the standard inner product (inherited from the ambient space) by a constant factor of $1/2$,

$$v_1^\top v_2 = \frac{1}{2}\,\mathrm{tr}(\eta_1^\top \eta_2). \tag{8.4}$$

8.2 Problem Formulation

Rotation Averaging. Recall the setup of (collaborative) rotation averaging from Section 2.2.1. The following shows an equivalent formulation of the optimization problem corresponding to rotation averaging,

Problem 8.1 (Rotation Averaging)**.**

$$\underset{R=(R_1,\ldots,R_n)\in\mathrm{SO}(d)^n}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \kappa_{ij}\varphi(R_i\widetilde{R}_{ij}, R_j). \tag{8.5}$$

For each edge $(i,j) \in \mathcal{E}$, $\kappa_{ij} > 0$ is the corresponding measurement weight. The function φ is defined as either the squared geodesic (8.6a) or chordal distance (8.6b),

$$\varphi(R_i\widetilde{R}_{ij}, R_j) \triangleq \begin{cases} \frac{1}{2}\left\|\mathrm{Log}(\widetilde{R}_{ij}^\top R_i^\top R_j)\right\|^2, & \text{(8.6a)} \\ \frac{1}{2}\left\|R_i\widetilde{R}_{ij} - R_j\right\|_F^2. & \text{(8.6b)} \end{cases}$$

Note that Problem 8.1 is equivalent to Problem 2.1 up to a constant factor of $1/2$ in the cost function that we include for convenience. The constant factor is inconsequential as it does not alter the solution to the optimization problem.

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

Problem 8.2 (Translation Estimation)**.**

$$\underset{t=(t_1,\ldots,t_n)\in\mathbb{R}^{d\times n}}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \frac{\tau_{ij}}{2}\left\|t_j - t_i - \widehat{t}_{ij}\right\|_2^2. \tag{8.7}$$

Note that (8.7) is a linear least squares problem. Similar to rotation averaging, (8.7) can be modeled using the undirected measurement graph $G = (\mathcal{V}, \mathcal{E})$, where vertex i represents the translation variable $t_i \in \mathbb{R}^d$ to be estimated, and edge $(i,j) \in \mathcal{E}$ represents the relative translation measurement $\widehat{t}_{ij} \in \mathbb{R}^d$. Lastly, $\tau_{ij} > 0$ is the positive weight associated with measurement $(i,j) \in \mathcal{E}$.

Initialization for PGO. In the context of collaborative SLAM, rotation averaging and translation estimation (Problem 8.2) can be combined to provide accurate *initialization* for PGO [31]. Recall the PGO formulation in Problem 2.2,

$$\underset{\substack{R=(R_1,\ldots,R_n)\in\mathrm{SO}(d)^n,\\ t=(t_1,\ldots,t_n)\in\mathbb{R}^{d\times n}}}{\text{minimize}} \sum_{(i,j)\in\mathcal{E}} \kappa_{ij}\left\|R_i\widetilde{R}_{ij} - R_j\right\|_F^2 + \sum_{(i,j)\in\mathcal{E}} \tau_{ij}\left\|t_j - t_i - R_i\widetilde{t}_{ij}\right\|_2^2. \tag{8.8}$$

In (8.8), $R_i \in \mathrm{SO}(d)$ and $t_i \in \mathbb{R}^d$ are rotation matrices and translation vectors to be estimated, $\widetilde{R}_{ij} \in \mathrm{SO}(d)$ and $\widetilde{t}_{ij} \in \mathbb{R}^d$ are noisy relative rotation and translation

measurements, and $\kappa_{ij}, \tau_{ij} > 0$ are constant measurement weights. Notice that in (8.8), the first sum of terms is equivalent to rotation averaging (Problem 8.1) under the chordal distance. Furthermore, given fixed rotation estimates $\widehat{R} \in \mathrm{SO}(d)^n$, the second sum of terms is equivalent to translation estimation (Problem 8.2) where each $\widehat{t}_{ij}$ in (8.7) is given by $\widehat{t}_{ij} = \widehat{R}_i \widetilde{t}_{ij}$. In both cases, the equivalence is up to a multiplying factor of $1/2$, but this is inconsequential since it does not change solutions to the optimization problems. Following Carlone *et al.* [31], we use these observations to initialize PGO in a two-stage process. The first stage initializes the rotation variables using the proposed rotation averaging solver (Section 8.4.2). Given the initial rotation estimates, the second stage initializes the translations using the proposed translation estimation solver (Section 8.4.3). We note that this initialization scheme does not have theoretical guarantees with respect to the full PGO problem. However, we still demonstrate its practical value through our experiments.

8.3 Laplacian Systems Arising from Rotation Averaging and Translation Estimation

In this section, we show that for rotation averaging (Problem 8.1) and translation estimation (Problem 8.2), their Hessian matrices are closely related to the Laplacians of suitably weighted graphs. The theoretical relations we establish in this section pave the way for designing fast and communication-efficient solvers in Section 8.4.

8.3.1 Rotation Averaging

To solve rotation averaging (Problem 8.1), we resort to an iterative Riemannian optimization framework. Before proceeding, however, one needs to be careful of the inherent *gauge-symmetry* of rotation averaging: in (8.5), note that left multiplying each rotation $R_i \in \mathrm{SO}(d), i \in [n]$ by a common rotation $S \in \mathrm{SO}(d)$ does not change the cost function. As a result, each solution $R = (R_1, \ldots, R_n) \in \mathrm{SO}(d)^n$ actually

corresponds to an *equivalence class* of solutions in the form of,

$$[R] = \{(SR_1, \ldots, SR_n),\ S \in \mathrm{SO}(d)\}. \tag{8.9}$$

The equivalence relation (8.9) shows that rotation averaging is actually an optimization problem defined over a *quotient manifold* $\mathcal{M} = \overline{\mathcal{M}}/\sim$, where $\overline{\mathcal{M}} = \mathrm{SO}(d)^n$ is called the total space and $\sim$ denotes the equivalence relation underlying (8.9); see [18, Chapter 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 $\mathcal{M}$ due to its superior convergence rate. The Newton update can be derived by considering a local perturbation of the cost function. Specifically, let $R = (R_1, \ldots, R_n) \in \mathrm{SO}(d)^n$ be our current rotation estimates. For each rotation matrix R_i, we seek a local correction to it in the form of $\mathrm{Exp}(v_i)R_i$, where $v_i \in \mathbb{R}^p$ is some vector to be determined and $\mathrm{Exp}(\cdot)$ is defined in (2.7). In (8.5), replacing each R_i with its correction $\mathrm{Exp}(v_i)R_i$ leads to the following local approximation[1] of the optimization problem,

$$\min_{v \in \mathbb{R}^{pn}} \quad h(v; R) \triangleq \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \varphi(\mathrm{Exp}(v_i)R_i\widetilde{R}_{ij}, \mathrm{Exp}(v_j)R_j). \tag{8.10}$$

In (8.10), the overall vector $v \in \mathbb{R}^{pn}$ is formed by concatenating all v_i's. Compared to (8.5), the optimization variable in (8.10) becomes the vector v and the rotations R are treated as fixed. Furthermore, we note that the quotient structure of Problem 8.1 gives rise to the following *vertical space* [18, Chapter 9.4] that summarizes all directions of

[1]The approximation defined in (8.10) is exactly the *pullback function* considered in Riemannian optimization [18, page 55], but expressed using a particular basis of the tangent space. We provide an explanation for the case of $d = 3$ (the case of $d = 2$ is simpler). Conventionally, the pullback is defined by considering a right perturbation of each rotation variable in the form of $R_i \,\mathrm{Exp}(v_i')$. In this work, we instead consider the *left* perturbation given by $\mathrm{Exp}(v_i)R_i$. However, using the adjoint operator on SO(3), we see that $\mathrm{Exp}(v_i)R_i = R_i \,\mathrm{Exp}(R_i^\top v_i)$, *i.e.*, $v_i' = R_i^\top v_i$. This shows that the basis we consider (corresponding to left perturbation) and the conventional basis (corresponding to right perturbation) only differ by an orthogonal transformation. In this work, we consider the left perturbation instead of the right perturbation, since the resulting Hessian has a particularly interesting relationship with the graph Laplacian matrix, as shown in Theorem 8.1.

change along which (8.10) is invariant,

$$\mathcal{N} = \text{image}(1_n \otimes I_p) \subset \mathbb{R}^{pn}. \tag{8.11}$$

Intuitively, $\mathcal{N}$ captures the action of any global left 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 (8.10) remains constant. Let us denote the gradient and Hessian of (8.10) as follows,

$$\overline{g}(R) \triangleq \nabla h(v; R)|_{v=0}, \quad \overline{H}(R) \triangleq \nabla^2 h(v; R)|_{v=0}. \tag{8.12}$$

Our notations $\overline{g}(R)$ and $\overline{H}(R)$ serve to emphasize that the gradient and Hessian are defined in the total space $\overline{\mathcal{M}}$ and depend on the current rotation estimates R. In [18, Chapter 9.12], it is shown that executing the Newton update on the quotient manifold amounts to finding the minimum norm solution of the following linear system,

$$\underbrace{(P_H \, \overline{H}(R) \, P_H)}_{H(R)} v = -\overline{g}(R), \tag{8.13}$$

where P_H is the orthogonal projection onto the *horizontal space* $\mathcal{H}$, defined as the orthogonal complement of the vertical space, *i.e.*, $\mathcal{H} \triangleq \mathcal{N}^{\perp}$. We note that P_H is symmetric, and so is $H(R)$. Furthermore, it holds that $\overline{g}(R) = P_H \overline{g}(R)$, which follows from known results on optimization over quotient manifolds (see Remark 8.2 for details). Intuitively, including P_H in (8.13) 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 (8.13) and the Laplacian of the underlying graph.

Theorem 8.1 (Local Hessian Approximation for Rotation Averaging)**.** *Let* $\underline{R} \in \text{SO}(d)^n$ *denote the set of ground truth rotations from which the noisy measurements* $\widetilde{R}_{ij}$ *are generated according to* (2.11)*. For any* $\delta \in (0, 1/2)$*, there exist constants* $\bar{\theta}, r > 0$ *such that if,*

$$\mathbf{d}(\widetilde{R}_{ij}, \underline{R}_i^{\top} \underline{R}_j) \leq \bar{\theta}, \ \forall (i,j) \in \mathcal{E}, \tag{8.14}$$

then for all $R \in B_r(R^\star) = \{R \in \mathrm{SO}(d)^n : \mathbf{d}(R, R^\star) < r\}$ *where* $R^\star \in \mathrm{SO}(d)^n$ *is a global minimizer of Problem 8.1, it holds that,*

$$H(R) \approx_\delta L(G; w) \otimes I_p. \tag{8.15}$$

In (8.15)*,* $G = (\mathcal{V}, \mathcal{E})$ *is the measurement graph. For each edge* $(i, j) \in \mathcal{E}$*, its edge weight* w_{ij} *used in* (8.15) *is given by* $w_{ij} = \kappa_{ij}$ *for the squared geodesic distance cost* (8.6a)*, and* $w_{ij} = 2\kappa_{ij}$ *for the squared chordal distance cost* (8.6b)*.*

Before proceeding, we note that Theorem 8.1 directly implies the following bound on the Hessian $H(R)$.

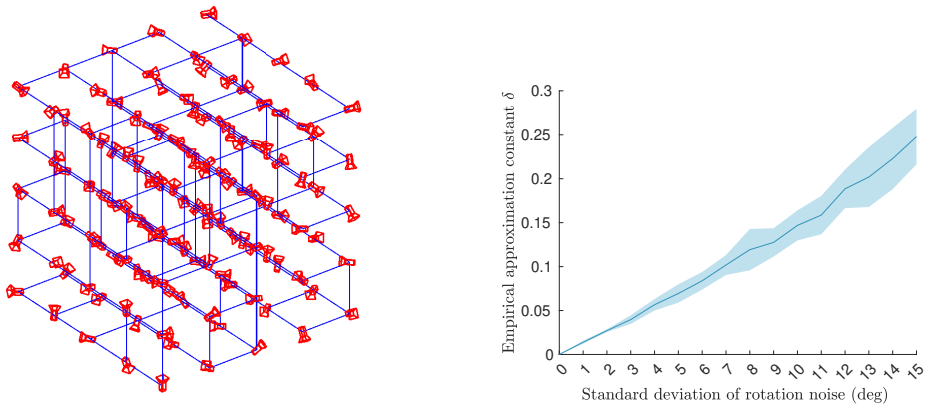
Corollary 8.1 (Local Hessian Bound and Condition Number for Rotation Averaging)**.** *Under the assumptions of Theorem 8.1, define constants* $\mu_H = e^{-\delta}\lambda_2(L(G; w))$ *and* $L_H = e^{\delta}\lambda_n(L(G; w))$*. Then for all* $R \in B_r(R^\star)$*,*

$$\mu_H P_H \preceq H(R) \preceq L_H P_H. \tag{8.16}$$

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

We prove Theorem 8.1 and Corollary 8.1 in Appendix D.2. Theorem 8.1 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 Figure 8-2, we perform numerical validation of this result using synthetic chordal rotation averaging problems defined over a 3D grid with 125 rotation variables (Figure 8-2a). With a probability of 0.3, we generate noisy relative measurements between pairs of nearby rotations, corrupted by increasing levels of Langevin noise [14, Appendix A]. At each noise level, we obtain the global minimizer $R^\star$ (global optimality is certified

[2]Currently, Theorem 8.1 only shows the existence of constants $\bar{\theta}, r > 0$ such that the approximation relation (8.15) holds. In a nutshell, this is because our proof is based on the following key relation that holds in the limit: if we define $\theta_{ij}(R) = \mathbf{d}(\widetilde{R}_{ij}, R_i^\top R_j)$ as the measurement residual of edge $(i, j) \in \mathcal{E}$ at a solution $R \in \mathrm{SO}(d)^n$, then we can show that $H(R) \to L(G; w) \otimes I_p$ as $\theta_{ij}(R) \to 0$ for all $(i, j) \in \mathcal{E}$; see discussions around (D.55) in the appendix. While it would be interesting to derive explicit and accurate bounds for $\bar{\theta}$ and r (as a function of δ), this would require a substantial improvement to our current proof technique, which we leave for future work.



(a) Grid simulation

(b) Empirical values of δ

Figure 8-2: Empirical validation of the Hessian approximation relation in Theorem 8.1. (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 δ such that $H(R^\star) \approx_\delta L \otimes I_p$. We perform 20 random runs for each noise level. Solid line denotes the average value for δ and the surrounding shaded area shows one standard deviation.

using the approach in [13]) and numerically compute the smallest constant δ such that $H(R^\star) \approx_\delta L \otimes I_p$. Figure 8-2b shows the evolution of the empirical approximation constant δ as a function of noise level. In the special case when there is no noise, it can be shown that $H(R^\star) = L \otimes I_p$, and thus the empirical δ is zero. In general, the empirical value of δ 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 $R^\star$, as predicted by Theorem 8.1.

The result in Theorem 8.1 directly motivates an *approximate Newton method* that replaces the Hessian with its Laplacian approximation. Specifically, instead of solving (8.13), one solves the following *approximate Newton* system,

$$(L(G; w) \otimes I_p)\, v = -\overline{g}(R). \tag{8.17}$$

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 $V, B(R) \in \mathbb{R}^{n\times p}$,

$$V \triangleq \begin{bmatrix} v_1^\top \\ \vdots \\ v_n^\top \end{bmatrix}, \quad B(R) \triangleq \begin{bmatrix} -\overline{g}_1(R)^\top \\ \vdots \\ -\overline{g}_n(R)^\top \end{bmatrix}. \tag{8.18}$$

Using properties of the Kronecker product, we can show that (8.17) is equivalent to,

$$L(G; w)V = B(R). \tag{8.19}$$

Algorithm 8.1 shows the pseudocode of the approximate Newton algorithm. Compared to the original Newton's method, Algorithm 8.1 uses a *constant matrix* across all iterations, and hence could be significantly more computationally efficient by avoiding to re-compute and re-factorize the Hessian matrix at every iteration. For this reason, we believe that Algorithm 8.1 could be of independent interest for standard (centralized) rotation averaging. Furthermore, in Section 8.4, we show that Algorithm 8.1 admits communication-efficient extensions in multi-robot settings.

Remark 8.1 (Connections with prior work)**.** Theorem 8.1 leverages prior theories developed by Tron [29] and Wilson *et al.* [126, 127] and extend them to cover rotation averaging under both geodesic and chordal distance metrics. Nasiri *et al.* [128] first developed Algorithm 8.1 for chordal rotation averaging using a Gauss-Newton formulation. In contrast, we motivate Algorithm 8.1 by proving the theoretical approximation relation between the Hessian and the graph Laplacian (Theorem 8.1). Lastly, the theoretical approximation relation we establish also allows us to prove local linear convergence for our methods.

Remark 8.2 (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 $\mathcal{N}$ is the vertical space defined in (8.11). Furthermore, in [18, Chapter 9.8], it is shown that $\overline{g}(R) \perp \mathcal{N}$. Thus, we conclude that $\overline{g}(R) \in \text{image}(L(G; w) \otimes I_p)$, *i.e.*, the linear system (8.17) and its equivalent matrix form (8.19) are always feasible. In fact, the system is singular and hence admits infinitely many solutions. Similar to

Algorithm 8.1 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 $R_i^{k+1} = \text{Exp}(v_i^k)R_i^k$, for all $i \in [n]$.
4: **end for**

the original Newton's method on quotient manifold, we will select the minimum norm solution v which guarantees that $v \perp \mathcal{N}$.

8.3.2 Translation Estimation

Unlike rotation averaging, translation estimation (Problem 8.2) is a convex linear least squares problem. In particular, it can be shown that Problem 8.2 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 τ_{ij} in Problem 8.2. Denote $M_t = \begin{bmatrix} t_1 & \ldots & t_n \end{bmatrix}^\top \in \mathbb{R}^{n \times d}$ as the matrix where each row corresponds to a translation vector to be estimated. One can show that the optimal translations are solutions of,

$$L(G; \tau)M_t = B_t, \tag{8.20}$$

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 (8.20) is always feasible; see [14, Appendix B.2] for details. To conclude this section, we note that similar to rotation averaging, translation estimation (Problem 8.2) 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 $M_t = M_t' + 1_n c^\top$ where $1_n \in \mathbb{R}^n$ is the vector of all ones and $c \in \mathbb{R}^d$ is some constant vector.

8.4 Algorithms and Performance Guarantees

In Section 8.3, we have shown that *Laplacian systems* naturally arise when solving the rotation averaging and translation estimation problems; see (8.19) and (8.20), respectively. Recall that we seek to find the solution $X \in \mathbb{R}^{n\times p}$ to a linear system of the form,

$$LX = B, \tag{8.21}$$

where $L \in \mathcal{S}_+^n$ is the Laplacian of the multi-robot measurement graph (see Figure 8-1a), and each column of $B \in \mathbb{R}^{n\times p}$ is in the image of L so that (8.21) is always feasible. In Section 8.4.1, we develop a communication-efficient solver for (8.21) under the server-client architecture. Then, in Section 8.4.2 and Section 8.4.3, we use the developed solver to design communication-efficient algorithms for collaborative rotation averaging and translation estimation, and establish convergence guarantees for both cases. Lastly, in Section 8.4.4, we present extension to outlier-robust estimation based on GNC.

8.4.1 A Collaborative Laplacian Solver with Spectral Sparsification

We propose to solve (8.21) using the *domain decomposition* framework [67, Chapter 14], which has been utilized in earlier works such as DDF-SAM [55–57] to solve collaborative SLAM problems. This is motivated by the fact that in the multi-robot measurement graph with m robots, there is a natural disjoint partitioning of the vertex set $\mathcal{V}$:

$$\mathcal{V} = \mathcal{V}_1 \uplus \ldots \uplus \mathcal{V}_m, \tag{8.22}$$

where $\mathcal{V}_\alpha$ contains all vertices (variables) of robot $\alpha \in [m]$. Furthermore, $\mathcal{V}_\alpha$ can be partitioned as $\mathcal{V}_\alpha = \mathcal{F}_\alpha \uplus \mathcal{C}_\alpha$ where $\mathcal{C}_\alpha$ denotes all separator (interface) vertices and $\mathcal{F}_\alpha$ denotes all interior vertices of robot α. In multi-robot SLAM, the separators are given by the set of variables that have inter-robot measurements; see Figure 8-1a. Note that given the set of all separators $\mathcal{C} = \mathcal{C}_1 \uplus \ldots \uplus \mathcal{C}_m$, robots' interior vertices

$\mathcal{F}_\alpha$ become disconnected from each other. The natural vertex partitioning in (8.22) further gives rise to a disjoint partitioning of the edge set,

$$\mathcal{E} = \mathcal{E}_1 \uplus \ldots \uplus \mathcal{E}_m \uplus \mathcal{E}_c. \tag{8.23}$$

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 Figure 8-1a, 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 Figure 8-1a.

In domain decomposition, we adopt a variable ordering in which the interior nodes $\mathcal{F} = \mathcal{F}_1 \uplus \ldots \uplus \mathcal{F}_m$ appear before the separators $\mathcal{C} = \mathcal{C}_1 \uplus \ldots \uplus \mathcal{C}_m$. With this variable ordering, the Laplacian system (8.21) can be rewritten as,

$$\begin{bmatrix} L_{11} & & & L_{1c} \\ & \ddots & & \vdots \\ & & L_{mm} & L_{mc} \\ L_{c1} & \ldots & L_{cm} & L_{cc} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_m \\ X_c \end{bmatrix} = \begin{bmatrix} B_1 \\ \vdots \\ B_m \\ B_c \end{bmatrix}. \tag{8.24}$$

For $\alpha \in [m]$, X_α and B_α denote the rows of X and B in (8.21) that correspond to robot α's interior variables $\mathcal{F}_\alpha$. On the other hand, we treat separators *from all robots* as a single block $\mathcal{C} = \mathcal{C}_1 \uplus \ldots \uplus \mathcal{C}_m$, and use X_c and B_c to denote the rows of X and B that correspond to the separator block $\mathcal{C}$.

Remark 8.3 (Computation of (8.24) under the server-client architecture)**.** Under the server-client architecture we consider, the overall Laplacian system (8.24) is stored distributedly across the robots (clients) and the server. Specifically, since each robot α knows the subgraph induced by its own vertices $\mathcal{V}_\alpha$ (*e.g.*, in Figure 8-1a, robot 2 knows all edges incident to the blue vertices), it independently computes and stores its Laplacian blocks $L_{\alpha\alpha}$ and $L_{\alpha c}$. Similarly, each robot α also independently computes and stores the block B_α. Meanwhile, we assume that the blocks defined over separators L_{cc} and B_c are handled by the central server that performs additional computations.

In (8.24), the special "arrowhead" sparsity pattern motivates us to first solve the *reduced system* defined over the separators,

$$\underbrace{\left(L_{cc} - \sum_{\alpha\in[m]} L_{c\alpha}L_{\alpha\alpha}^{-1}L_{\alpha c}\right)}_{S=\mathrm{Sc}(L,\mathcal{F})} X_c = \underbrace{B_c - \sum_{\alpha\in[m]} L_{c\alpha}L_{\alpha\alpha}^{-1}B_\alpha}_{U}. \tag{8.25}$$

In the following, let us define $U_\alpha \triangleq L_{c\alpha}L_{\alpha\alpha}^{-1}B_\alpha$ for each robot $\alpha \in [m]$. Then, the matrix on the right-hand side of (8.25) can be written as,

$$U \triangleq B_c - \sum_{\alpha\in[m]} U_\alpha. \tag{8.26}$$

Meanwhile, the matrix S defined on the left-hand side of (8.25) is the Schur complement resulting from eliminating all interior nodes $\mathcal{F}$ from the full Laplacian matrix L, denoted as $S = \mathrm{Sc}(L, \mathcal{F})$. The next lemma shows S is the sum of multiple smaller Laplacian matrices.

Lemma 8.1. *For each robot $\alpha \in [m]$, define $G_\alpha = (\mathcal{F}_\alpha \uplus \mathcal{C}, \mathcal{E}_\alpha)$ as its local graph induced by its interior edges $\mathcal{E}_\alpha$. Let S_α be the matrix resulting from eliminating robot α's interior vertices $\mathcal{F}_\alpha$ from the Laplacian of G_α,* i.e., *$S_\alpha = \mathrm{Sc}(L(G_\alpha), \mathcal{F}_\alpha)$. 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* (8.25) *can be written as,*

$$S = L(G_c) + \sum_{\alpha\in[m]} S_\alpha. \tag{8.27}$$

Lemma 8.1 is proved in Appendix D.3.1. Since Laplacian matrices are closed under Schur complements [141, Fact 4.2], each S_α defined in Lemma 8.1 is also a Laplacian matrix.[3] Furthermore, as a result of Remark 8.3, each robot α can independently

[3]In Lemma 8.1, we can technically define $G_\alpha = (\mathcal{F}_\alpha \uplus \mathcal{C}_\alpha, \mathcal{E}_\alpha)$ since $\mathcal{E}_\alpha$ only involves robot α's vertices. However, we choose to involve all separators and define $G_\alpha = (\mathcal{F}_\alpha \uplus \mathcal{C}, \mathcal{E}_\alpha)$, where any separator from $\mathcal{C} \setminus \mathcal{C}_\alpha$ simply does not have any edges. This is done for notation simplicity, so that after eliminating $\mathcal{F}_\alpha$ from G_α, the resulting S_α matrix is defined over all separators and thus can be added together as in (8.27).

Algorithm 8.2 SPARSIFIED SCHUR COMPLEMENT

function $\widetilde{S}$ = SPARSIFIEDSCHURCOMPLEMENT(L, ϵ)
  for each robot α **in parallel do**
    Compute a sparse approximation $\widetilde{S}_\alpha$ such that $\widetilde{S}_\alpha \approx_\epsilon S_\alpha$.
    Upload $\widetilde{S}_\alpha$ to the server.
  end for
  Server computes and stores $\widetilde{S} = L(G_c) + \sum_{\alpha\in[m]} \widetilde{S}_\alpha$.
end function

Algorithm 8.3 SPARSIFIED LAPLACIAN SOLVER

function X = SPARSIFIEDLAPLACIANSOLVER(L, B, $\widetilde{S}$)
  for each robot α **in parallel do**
    Compute $U_\alpha = L_{c\alpha} L_{\alpha\alpha}^{-1} B_\alpha$.
    Upload U_α to the server.
  end for
  Server collects U_α and computes $U = B_c - \sum_{\alpha\in[m]} U_\alpha$.
  Server solves $\widetilde{S} X_c = U$ (where $\widetilde{S}$ is obtained from Algorithm 8.2), and broadcasts solution X_c to all robots.
  for each robot α **in parallel do**
    Compute interior solution $X_\alpha = L_{\alpha\alpha}^{-1} \left(B_\alpha - L_{\alpha c} X_c\right)$.
  end for
end function

compute $S_\alpha = \mathrm{Sc}(L(G_\alpha), \mathcal{F}_\alpha)$ and $U_\alpha = L_{c\alpha} L_{\alpha\alpha}^{-1} B_\alpha$. This observation motivates a method in which robots first transmit their S_α and U_α to the server in parallel. Upon collecting S_α and U_α from all robots, the server can then form S using (8.27) and U using (8.26). It then solves the linear system $SX_c = U$ (8.25) 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,

$$X_\alpha = L_{\alpha\alpha}^{-1} \left(B_\alpha - L_{\alpha c} X_c\right). \tag{8.28}$$

The aforementioned method is a multi-robot 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 α to transmit its Schur complement matrix S_α that is potentially dense. This issue is illustrated in Figure 8-1b, where for robot 2 (blue) its S_α 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_α is itself a graph Laplacian, and (ii) graph Laplacians admit *spectral sparsifications* [135], *i.e.*, for a given approximation threshold $\epsilon > 0$, one can compute a sparse Laplacian $\widetilde{S}_\alpha$ such that $\widetilde{S}_\alpha \approx_\epsilon S_\alpha$. Generally, a larger value of ϵ leads to a sparser $\widetilde{S}_\alpha$. In this work, we implement the method of Spielman and Srivastava [136] that sparsifies S_α 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 $\widetilde{S}_\alpha$ 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_α. Appendix D.1 provides the complete description and pseudocode of the sparsification algorithm. Figure 8-1c illustrates a spectral sparsification for robot 2's dense reduced graph. In the proposed method, each robot transmits its sparse approximation $\widetilde{S}_\alpha$ instead of the original Schur complement S_α. By summing together these $\widetilde{S}_\alpha$ matrices, the server can obtain a sparse approximation to the original dense Schur complement S; see Algorithm 8.2. Then, we can follow the same procedure as standard domain decomposition to obtain an approximate solution to the Laplacian system (8.21); see Algorithm 8.3. Specifically, the server first solves an *approximate* reduced system using $\widetilde{S}$ obtained from Algorithm 8.2 (line 7). Then, the interior solution for each robot is recovered using back-substitution (line 9).

Together, Algorithms 8.2 and 8.3 provide a parallel procedure for computing an approximate solution to the original Laplacian system (8.21) 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 8.2 (Approximation guarantees of Algorithms 8.2 and 8.3)**.** *Given a Laplacian system $LX = B$, Algorithms 8.2 and 8.3 together return a solution $\widetilde{X} \in \mathbb{R}^{n \times p}$*

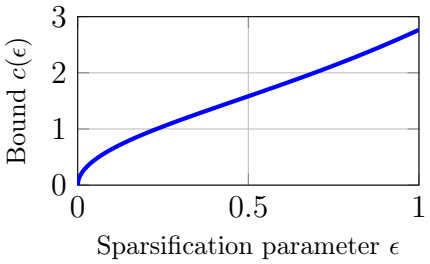


Figure 8-3: Visualization of $c(\epsilon)$ in Theorem 8.2.

such that $\widetilde{L}\widetilde{X} = B$*, where* $\widetilde{L} \in \mathcal{S}_+^n$ *satisfies,*

$$\widetilde{L} \approx_\epsilon L. \tag{8.29}$$

Furthermore, let $X^\star \in \mathbb{R}^{n\times p}$ *be an exact solution to the input linear system,* i.e., $LX^\star = B$*. It holds that,*

$$\left\|\widetilde{X} - X^\star\right\|_L \leq c(\epsilon)\left\|X^\star\right\|_L, \tag{8.30}$$

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

$$c(\epsilon) = \sqrt{1 + e^{2\epsilon} - 2e^{-\epsilon}}. \tag{8.31}$$

We prove Theorem 8.2 in Appendix D.3.2. We have shown that the approximate solution $\widetilde{X}$ produced by Algorithms 8.2 and 8.3 remains close to the exact solution $X^\star$ when measured using the "norm" induced by the original Laplacian L.[4] Furthermore, the quality of the approximation is controlled by the sparsification parameter ϵ through the function $c(\epsilon)$ visualized in Figure 8-3. Note that when $\epsilon = 0$, sparsification is effectively skipped and robots transmit the original dense matrices S_α. In this case, we have $c(\epsilon) = 0$ and the solution $\widetilde{X}$ produced by our methods is exact, *i.e.*, $L\widetilde{X} = B$. Meanwhile, by increasing ϵ, our methods smoothly trade off accuracy with communication efficiency.

[4]The reader might question the use of $\|\cdot\|_L$ in (8.30) because the Laplacian L is singular. Indeed, due to the singularity of L, $||X^\star - \widetilde{X}||_L$ ignores any component of $X^\star - \widetilde{X}$ that lives on the kernel of L, which is spanned by the vector of all ones 1_n. However, this does not create a problem for us since we only seek to compare $X^\star$ and $\widetilde{X}$ *when considering both as solutions to the Laplacian system* $LX = B$, and using $\|\cdot\|_L$ naturally eliminates any difference on $\ker(L)$ that is inconsequential.

Remark 8.4 (Connections with existing Laplacian solvers [141, 142])**.** Our collaborative Laplacian solver (Algorithms 8.2 and 8.3) is inspired by the centralized solvers developed in [141, 142] 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 multi-robot setting to achieve communication efficiency. Furthermore, in Section 8.4.2, we apply our Laplacian solver on the non-convex Riemannian optimization problem underlying rotation averaging, and establish provable convergence guarantees for the resulting Riemannian optimization algorithm.

Remark 8.5 (Communication efficiency of Algorithms 8.2 and 8.3)**.** We discuss the communication costs of Algorithms 8.2 and 8.3 under the server-client architecture. Denote the number of separators in the measurement graph as $|\mathcal{C}|$. In Algorithm 8.2, each robot uploads the sparsified matrix $\widetilde{S}_\alpha$ to the server (line 4), which is guaranteed to have $O(|\mathcal{C}|\log|\mathcal{C}|)$ entries [136]. Consequently, Algorithm 8.2 incurs a total *upload* cost of $O(m|\mathcal{C}|\log|\mathcal{C}|)$, where m is the number of robots. In Algorithm 8.3, robots upload their block vectors U_α in parallel (line 4) and the server broadcasts back the solution X_c (line 7). Since both U_α and X_c have a dimension of $|\mathcal{C}|$-by-p (where $p = \dim \mathrm{SO}(d)$ is constant), Algorithm 8.3 uses $O(m|\mathcal{C}|)$ communication in both *upload* and *download* stages.

8.4.2 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 8.1, where each iteration solves a Laplacian system $LV = B(R)$. In the multi-robot setting, we can use the solver developed in Section 8.4.1 to obtain an approximate solution to this system. Algorithm 8.4 shows the pseudocode. First, an initial guess R^0 is computed (line 1). Then, at line 2, robots first form the approximate Schur complement $\widetilde{S}$ using SparsifiedSchurComplement (Algorithm 8.2). 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

Algorithm 8.4 COLLABORATIVE ROTATION AVERAGING

1: Initialize rotation estimates R^0.
2: $\widetilde{S} = \text{SPARSIFIEDSCHURCOMPLEMENT}(L, \epsilon)$.
3: **for** iteration $k = 0, 1, \ldots$ **do**
4: // Distributed computation of $B(R^k)$
5: Server computes $B(R^k)_c$ that corresponds to all separators.
6: **for** each robot α **in parallel do**
7: Compute $B(R^k)_\alpha$ that corresponds to interior $\mathcal{F}_\alpha$.
8: **end for**
9: // Single round of communication to compute V^k
10: Solve $V^k = \text{SPARSIFIEDLAPLACIANSOLVER}(L, B(R^k), \widetilde{S})$.
11: // Distributed updates of all rotation variables
12: **for** each robot α **in parallel do**
13: Update iterates by $R_i^{k+1} = \text{Exp}(v_i^k) R_i^k$, for each rotation variable R_i owned by robot α.
14: **end for**
15: **end for**

from Remark 8.3 that the overall $B(R)$ is divided into multiple blocks,

$$B(R) = \begin{bmatrix} B(R)_1^\top & \ldots & B(R)_m^\top & B(R)_c^\top \end{bmatrix}^\top. \tag{8.32}$$

In our algorithm, each robot $\alpha \in [m]$ computes the block $B(R)_\alpha$ corresponding to its interior variables $\mathcal{F}_\alpha$, and the server computes the block $B(R)_c$ corresponding to all separators. At the second step (line 10), robots collaboratively solve for the update vector V^k by calling SPARSIFIEDLAPLACIANSOLVER (Algorithm 8.3). 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 8.4 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.*, $\mathbf{d}(R^{k+1}, R^\star) \leq \gamma\, \mathbf{d}(R^k, R^\star)$ where k is the iteration number, $\gamma \in (0, 1)$ is a constant, and $R^\star$ 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

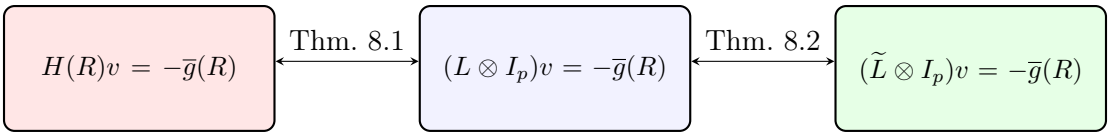


Figure 8-4: Intuitions behind the convergence rate in Theorem 8.3. Recall from Theorem 8.1 that under bounded measurement noise, the original Newton system (left box) is locally δ-approximated by a linear system specified by a Laplacian L (middle box). In addition, in Theorem 8.2 we have shown that our distributed Laplacian solver approximates L with $\widetilde{L}$ where $L \approx_\epsilon \widetilde{L}$ (right box). The composition of the two approximation relations thus gives $H(R) \approx_{\delta+\epsilon} (\widetilde{L} \otimes I_p)$, which intuitively explains why (8.34) depends on a function of $\delta + \epsilon$.

solution R^∞ that is only equivalent to $R^\star$ *up to a global rotation*, *i.e.*,

$$(SR_1^\infty, \ldots, SR_n^\infty) = (R_1^\star, \ldots, R_n^\star), \text{ for some } S \in \mathrm{SO}(d), \tag{8.33}$$

and as a result $\mathbf{d}(R^\infty, R^\star) \neq 0$ in general. Fortunately, this issue can be resolved using the machinery of Riemannian quotient manifolds. Instead of measuring the distance on the total space $\mathbf{d}(R^k, R^\star)$, we will compute the distance between the underlying equivalence classes $\mathbf{d}([R^k], [R^\star])$. We note that $\mathbf{d}([R^k], [R^\star])$ is well-defined since a quotient manifold inherits the Riemannian metric from its total space [18, Chapter 9]. Equipped with this distance metric, we are ready to formally state the convergence result for Algorithm 8.4.

Theorem 8.3 (Convergence rate of Algorithm 8.4)**.** *Define* $\gamma(x) = 2\sqrt{\kappa_H} c(x)$ *where* $\kappa_H = L_H/\mu_H$ *is the condition number in Corollary 8.1 and* $c(\cdot)$ *is defined in* (8.31)*. Under the assumptions of Theorem 8.1, suppose* ϵ *is selected such that* $\gamma(\delta + \epsilon) < 1$*. In addition, suppose at each iteration* k*, the update vector* $v^k \in \mathbb{R}^{pn}$ *is orthogonal to the vertical space,* i.e.*,* $v^k \perp \mathcal{N}$*. Let* $R^\star$ *be an optimal solution to Problem 8.1. There exists* $r' > 0$ *such that for any* R^0 *where* $\mathbf{d}([R^0], [R^\star]) < r'$*, Algorithm 8.4 generates an infinite sequence* $\{R^k\}$ *where the corresponding sequence of equivalence classes* $[R^k]$ *converges linearly to* $[R^\star]$*. Furthermore, the convergence rate factor is,*

$$\limsup_{k\to\infty} \frac{\mathbf{d}([R^{k+1}], [R^\star])}{\mathbf{d}([R^k], [R^\star])} = \gamma(\delta + \epsilon). \tag{8.34}$$

We prove Theorem 8.3 in Appendix D.4.2. Theorem 8.3 shows that using the

distance metric on the quotient manifold, Algorithm 8.4 locally converges to the global minimizer at a linear rate.[5] Figure 8-4 provides intuitions behind the convergence rate in (8.34). Recall that δ appears in Theorem 8.1 where we show $H(R) \approx_\delta (L \otimes I_p)$ under bounded measurement noise. On the other hand, ϵ is the parameter for spectral sparsification and is controlled by the user. In Theorem 8.2, we showed that our methods transform the input Laplacian L into an approximation $\widetilde{L}$ such that $L \approx_\epsilon \widetilde{L}$. The composition of the two approximation relations thus gives $H(R) \approx_{\delta+\epsilon} (\widetilde{L} \otimes I_p)$, which intuitively explains why the convergence rate depends on a function of $\delta + \epsilon$. Lastly, we note that while our theoretical convergence guarantees require $\gamma(\delta+\epsilon) < 1$, our experiments (Section 8.5) show that Algorithm 8.4 is not sensitive to the choice of ϵ and converges under a wide range of parameter settings.

Remark 8.6 (Communication efficiency of Algorithm 8.4)**.** In Algorithm 8.4, note that only a single call to SparsifiedSchurComplement (Algorithm 8.2) is needed, which incurs a total upload of $O(m|\mathcal{C}|\log|\mathcal{C}|)$; see Remark 8.5. In each iteration, a single call to SparsifiedLaplacianSolver (Algorithm 8.3) is made, which requires a single round of upload and download. Furthermore, by Remark 8.5, both upload and download costs are bounded by $O(m|\mathcal{C}|)$. Therefore, after $K > 0$ iterations, Algorithm 8.4 uses a total upload of $O(m|\mathcal{C}|\log|\mathcal{C}| + mK|\mathcal{C}|)$ and a total download of $O(mK|\mathcal{C}|)$. In particular, the terms that involve the number of iterations K scales *linearly* with the number of separators $|\mathcal{C}|$, which makes the algorithm very communication-efficient.

8.4.3 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 (8.20) as shown in Section 8.3.2. Specifically, we employ our collaborative Laplacian

[5] In Theorem 8.3, the orthogonality assumption $v^k \perp \mathcal{N}$ is needed to ensure that the update vector v^k corresponds to a valid tangent vector on the tangent space of the underlying quotient manifold; see Appendix D.4.2 for details. One can satisfy this assumption by projecting v^k to the horizontal space, which requires a single round of communication between the server and robots. However, in practice, we find that this has have negligible impact on the iterates and thus skip this step in our implementation.

Algorithm 8.5 COLLABORATIVE TRANSLATION ESTIMATION

1: Initialize translation estimates $M_t^0 = 0_{n\times d}$.
2: $\widetilde{S} = \text{SPARSIFIEDSCHURCOMPLEMENT}(L, \epsilon)$.
3: **for** iteration $k = 0, 1, \ldots$ **do**
4: // Distributed computation of E^k
5: Server computes E_c^k that corresponds to all separators.
6: **for** each robot α **in parallel do**
7: Compute E_α^k that corresponds to interior $\mathcal{F}_\alpha$.
8: **end for**
9: // Single round of communication to compute D^k
10: Solve $D^k = \text{SPARSIFIEDLAPLACIANSOLVER}(L, E^k, \widetilde{S})$.
11: // Distributed updates of all translations: $M_t^{k+1} = M_t^k + D^k$
12: **for** each robot α **in parallel do**
13: Update iterates by $t_i^{k+1} = t_i^k + (D_{[i,:]}^k)^\top$ for each translation variable t_i owned by robot α.
14: **end for**
15: **end for**

solver (Section 8.4.1) in an *iterative refinement* framework. Let $M_t^k \in \mathbb{R}^{n\times d}$ be our estimate for the translation variables at iteration k (in practice M_t^0 can simply be initialized at zero). We seek a correction D^k to M_t^k by solving the *residual system* corresponding to (8.20):

$$L(M_t^k + D^k) = B_t \iff LD^k = B_t - LM_t^k \triangleq E^k. \tag{8.35}$$

Observing that the system on the right-hand side of (8.35) is another Laplacian system in $L \equiv L(G;\tau)$, we can deploy our Laplacian solver to find an approximate solution D^k. Algorithm 8.5 shows the pseudocode, which shares many similarities with the proposed collaborative rotation averaging method Algorithm 8.4. 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 complexity; see Remark 8.6. The following theorem states the theoretical guarantees for Algorithm 8.5.

Theorem 8.4 (Convergence rate of Algorithm 8.5)**.** *Suppose ϵ is selected such that the constant $c(\epsilon)$ defined in* (8.31) *satisfies $c(\epsilon) < 1$. Let $M_t^\star$ be an optimal solution to Problem 8.2 and let M_t^k denote the solution computed by Algorithm 8.5 at iteration*

$k \geq 1$. *It holds that,*

$$\left\|M_t^k - M_t^\star\right\|_L \leq c(\epsilon)^k \left\|M_t^\star\right\|_L, \tag{8.36}$$

where $L \equiv L(G;\tau)$.

We prove Theorem 8.4 in Appendix D.4.3. Theorem 8.4 is simpler compared to its counterpart for rotation averaging (Theorem 8.3). The convergence rate (8.36) only depends on the sparsification parameter ϵ. Furthermore, since the translation estimation problem is convex, the convergence guarantee is *global* and holds for any initial guess.[6] While Theorem 8.4 requires $c(\epsilon) < 1$, our experiments show that Algorithm 8.5 is not sensitive to the choice of sparsification parameter ϵ and converges under a wide range of parameter settings.

8.4.4 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 multi-robot SLAM). In this section, we present an extension to *outlier-robust* optimization by embedding the developed solvers in the graduated non-convexity (GNC) framework [28, 189]. A similar extension has been developed in Chapter 6, but under a fully distributed architecture instead of the server-client architecture considered in this chapter. Specifically, we consider robust estimation using the truncated least squares (TLS) cost:[7]

$$\underset{x \in \mathcal{X}}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \rho^{\text{TLS}}(e_{ij}(x)). \tag{8.37}$$

In (8.37), $x \in \mathcal{X}$ 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 extension of rotation averaging (Problem 8.1), we define $x = (R_1, \ldots, R_n) \in \text{SO}(d)^n$, and $e_{ij}(x) =$

[6] In (8.36), the use of $\|\cdot\|_L$ naturally accounts for the global translation symmetry of Problem 8.2 (see Section 8.3.2). Specifically, since $\ker(L) = \text{image}(1_n)$, $||M_t^k - M_t^\star||_L$ disregards any difference between M_t^k and $M_t^\star$ that corresponds to a global translation.

[7] Other robust cost functions, such as the Geman McClure function, can also be used in the same framework; see [28].

$\sqrt{\kappa_{ij}/2}\,\mathbf{d}(R_i\widetilde{R}_{ij}, R_j)$ where $\mathbf{d}(\cdot,\cdot)$ is the geodesic or the chordal distance. For the robust extension of translation estimation (Problem 8.2), we define $x = (t_1, \ldots, t_n) \in \mathbb{R}^{d\times n}$ and $e_{ij}(x) = \sqrt{\tau_{ij}/2}\left\|t_j - t_i - \widehat{t}_{ij}\right\|$. Notice that $e_{ij}(x)$ is simply the square root of a single cost term in Problem 8.1 or Problem 8.2. Finally, $\rho^{\text{TLS}}(e) \triangleq \min(e^2, \overline{e}^2)$ denotes the TLS cost function, where $\overline{e}$ 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 $\overline{e}$.

To mitigate the non-convexity introduced by robust cost functions, GNC solves (8.37) by optimizing a sequence of easier (*i.e.*, less non-convex) surrogate functions ρ_μ^{TLS} that gradually converges to the original, highly non-convex cost function ρ^{TLS}. Here, μ is the control parameter and for the TLS function, it satisfies that (i) ρ_μ^{TLS} is convex for $\mu \to 0$, and (ii) ρ_μ^{TLS} recovers ρ^{TLS} for $\mu \to +\infty$; see [28, Example 2]. In practice, we initialize by setting $\mu \approx 0$, and gradually increase μ as optimization progresses. Furthermore, leveraging the Black-Rangarajan duality [189], each surrogate problem can be formulated as follows,

$$\underset{x\in\mathcal{X}, w_{ij}^{\text{GNC}}\in[0,1]}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} \left[w_{ij}^{\text{GNC}} e_{ij}^2(x) + \Phi_\mu(w_{ij}^{\text{GNC}})\right]. \tag{8.38}$$

In (8.38), w_{ij}^{GNC} is a mutable weight attached to the measurement error e_{ij}, and Φ_μ acts as a regularization term on the weight whose expression depends on the control parameter μ.

GNC leverages (8.38) by performing alternating updates on the model x and the weights w_{ij}^{GNC}, while simultaneously updating the control parameter μ. Specifically, each GNC outer iteration consists of three steps:

1. **Variable update**: optimize the surrogate problem (8.38) with respect to x, under fixed weights w_{ij}^{GNC}. Notice that this amounts to a standard weighted least squares problem,

$$\underset{x\in\mathcal{X}}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{E}} w_{ij}^{\text{GNC}} e_{ij}^2(x). \tag{8.39}$$

Algorithm 8.6 Outlier-robust rotation averaging with GNC

1: Initialize control parameter μ 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 8.4 under the server-client architecture.
4: **Weight update**: in parallel, server computes (8.40) for all inter-robot measurements $\mathcal{E}_c$, and each robot α computes (8.40) for its local measurements $\mathcal{E}_\alpha$.
5: **Parameter update**: in parallel, server and all robots updates the control parameter μ.
6: **end while**

2. **Weight update**: optimize the surrogate problem (8.38) with respect to all w_{ij}^{GNC}, under fixed model x. For TLS, the resulting w_{ij}^{GNC} has a *closed-form* solution,

$$w_{ij}^{\text{GNC}} \leftarrow \begin{cases} 0, & \text{if } e_{ij}^2 \in \left[\frac{\mu+1}{\mu}\bar{e}^2, +\infty\right], \\ \frac{\bar{e}}{e_{ij}}\sqrt{\mu(\mu+1)} - \mu, & \text{if } e_{ij}^2 \in \left[\frac{\mu}{\mu+1}\bar{e}^2, \frac{\mu+1}{\mu}\bar{e}^2\right], \\ 1, & \text{if } e_{ij}^2 \in \left[0, \frac{\mu}{\mu+1}\bar{e}^2\right], \end{cases} \tag{8.40}$$

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

3. **Parameter update**: update control parameter μ via $\mu \leftarrow 1.4\mu$ (recommended in [28, Remark 5]), and move on to the next surrogate problem.

Initially, all measurement weights are initialized at one.

Next, we show that our algorithms developed in this chapter can be used within GNC to perform outlier-robust optimization. Algorithm 8.6 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 (8.39) solved during the **variable update** step have identical forms as Problems 8.1 and 8.2. The only difference is that each measurement is now discounted by the GNC weight w_{ij}^{GNC}, as shown in (8.39). Therefore, we can use Algorithm 8.4 to perform the variable update for rotation averaging (line 3), and Algorithm 8.5 for translation estimation. Furthermore, the

weight update step can also be executed under the server-client architecture, where each robot α computes (8.40) for its local measurements $\mathcal{E}_\alpha$, and the server handles the inter-robot measurements $\mathcal{E}_c$; see line 4. Lastly, the server and all robots can in parallel perform the **parameter update** step by updating their local copies of the control parameter μ (line 5).

Remark 8.7 (Implementation details of GNC)**.** We discuss several implementation details for GNC.

- *Initialization.* In Chapter 6, we have observed that using an outlier-free initial guess when solving the variable update step is critical to ensure good performance. For multi-robot SLAM, we adopt the method described in Section 6.4 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.

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

- *Approximate optimization.* Recall that each outer iteration of GNC invokes Algorithm 8.4 or Algorithm 8.5 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 8.5.3). 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.

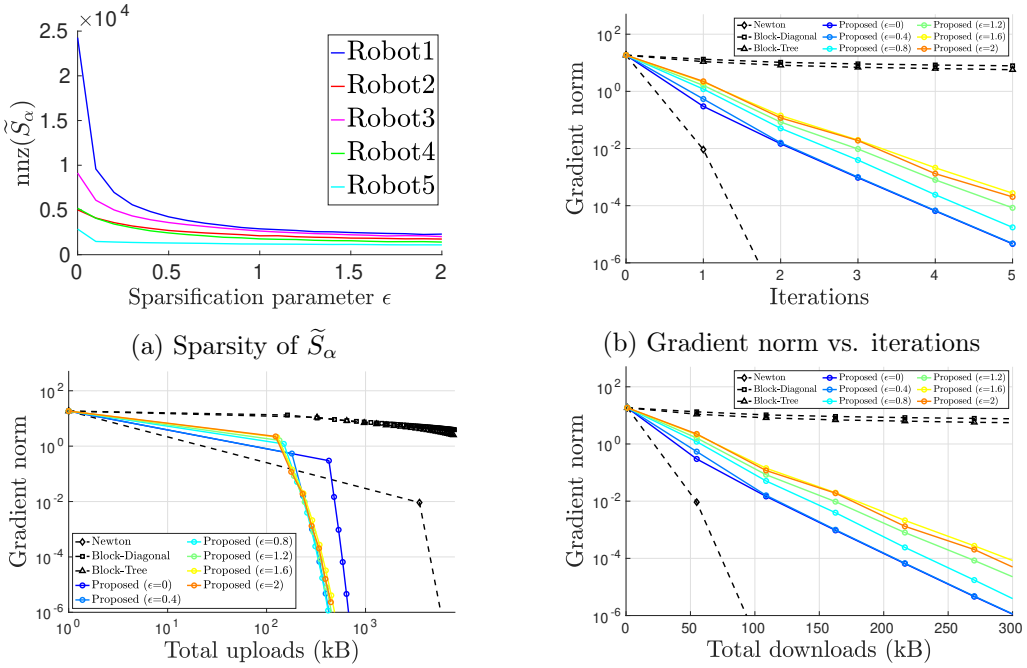
We conclude this subsection by noting that the linear convergence results (Theorems 8.3 and 8.4) we prove in this chapter only hold for the outlier-free case. Ex-

tending 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.

8.5 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 8.5.1 and 8.5.2 show evaluations using synthetic and benchmark datasets. Then, Section 8.5.3 and Section 8.5.4 demonstrate *outlier-robust* estimation using our approach on real-world collaborative SLAM and SfM problems. Lastly, Section 8.5.5 provides additional discussions on the performance of our approach in real-world problem instances. All algorithms are implemented in MATLAB and evaluated on a computer with an Intel i7-7700K CPU and 16 GB RAM.

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_{init} and f_{opt} denote the cost achieved by our initialization and the global minimizer, respectively. Lastly, 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 $\widehat{R} \in \mathrm{SO}(d)^n$ and the reference $R^{\text{ref}} \in \mathrm{SO}(d)^n$ (either global minimizer or ground truth) using the



(a) Sparsity of $\widetilde{S}_\alpha$

(b) Gradient norm vs. iterations

(c) Gradient norm vs. uploads

(d) Gradient norm vs. downloads

Figure 8-5: Evaluation of Algorithm 8.4 on the 5-robot rotation averaging problem from the Cubicle dataset. (a) For each robot α, we show the number of nonzero entries (nnz) in its sparsified matrix $\widetilde{S}_\alpha$ as a function sparsification parameter ϵ. (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.

orbit distance:

$$\mathrm{RMSE}(\widehat{R}, R^{\mathrm{ref}}) \triangleq \min_{S \in \mathrm{SO}(d)} \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left\| S\widehat{R}_i - R_i^{\mathrm{ref}} \right\|_F^2}. \tag{8.41}$$

Intuitively, (8.41) computes the root-mean-square error (RMSE) between two sets of rotations after alignment by a global rotation. The optimal alignment S in (8.41) has a closed-form expression; see [14, Appendix C.1]. Similarly, for translations, we report the RMSE between our solution and the reference after a global alignment.

8.5.1 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 8.4 using the distributed chordal initialization approach in [38], where the number of iterations is limited to 50. Our experiments mainly consider rotation averaging problems under the chordal distance metric.

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 8.4), by simulating a 5-robot problem using the Cubicle dataset. Recall that Algorithm 8.4 calls the SparsifiedSchurComplement procedure (Algorithm 8.2), which requires each robot α to transmit its sparsified matrix $\widetilde{S}_\alpha$. Figure 8-5a shows the number of nonzero entries in $\widetilde{S}_\alpha$ as a function of the sparsification parameter ϵ. Note that when $\epsilon = 0$, sparsification is effectively skipped and each robot transmits its exact S_α matrix that is potentially large and dense. In Figure 8-5a, this is reflected on robot 1 (blue curve) whose exact S_α matrix has more than 2×10^4 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 $\epsilon = 2$ creates a sparse $\widetilde{S}_\alpha$ with 2300 nonzero entries, which is much sparser than the original S_α.

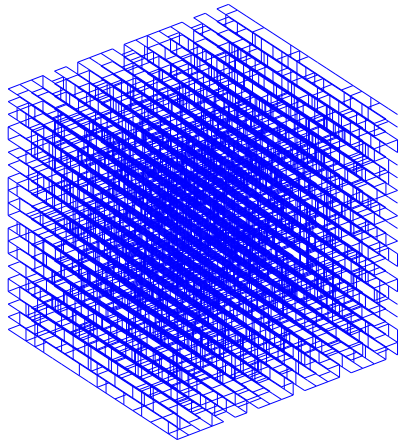
Next, we evaluate the convergence rate and communication efficiency of Algorithm 8.4 with varying sparsification parameter ϵ. We introduce three baseline methods for the purpose of comparison. The first baseline, called Newton in Figure 8-5, implements the exact Newton update using domain decomposition, where each robot computes and transmits the Schur complement of its local Hessian matrix to the server. Inspired by existing works [66, 68–70], we also implement two baselines that apply heuristic sparsification to Newton: in Block-Diagonal, each robot only transmits the diagonal blocks of its Hessian Schur complement (this strategy is also known as Jacobi preconditioning [67]), whereas in Block-Tree, each robot transmits both diagonal blocks and off-diagonal blocks that form a tree sparsity pattern. Figure 8-5b shows

the accuracy achieved by all methods (measured by norm of the Riemannian gradient) as a function of iterations. As expected, 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.[8] In contrast, our proposed method achieves fast convergence under a wide range of sparsification parameter ϵ. Furthermore, by varying ϵ, the proposed method provides a principled way to trade off convergence speed with communication efficiency.

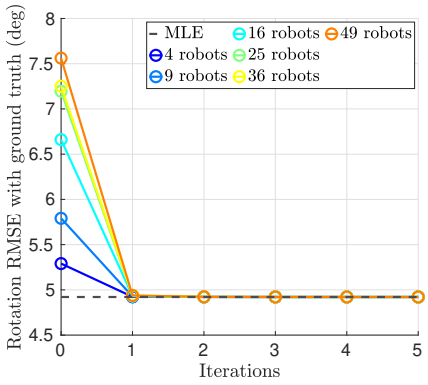
Figure 8-5c visualizes the accuracy as a function of total uploads to the server. Since both the Hessian and Laplacian matrices are symmetric, we only record the communication when uploading their upper triangular parts as sparse matrices. To convert the result to kilobyte (kB), we assume each scalar is transmitted in double precision. Our results show that the proposed method achieves the best communication efficiency under various settings of the sparsification parameter ϵ. Moreover, even without sparsification (*i.e.*, $\epsilon = 0$), the proposed method is still more communication-efficient than Newton. This result is due to the following reasons. First, since the Hessian matrix varies across iterations, 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 8.4. Second, 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 $p = \dim \mathrm{SO}(d)$ is the intrinsic dimension of the rotation group (for the 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.

Lastly, Figure 8-5d visualizes the accuracy as a function of total communication

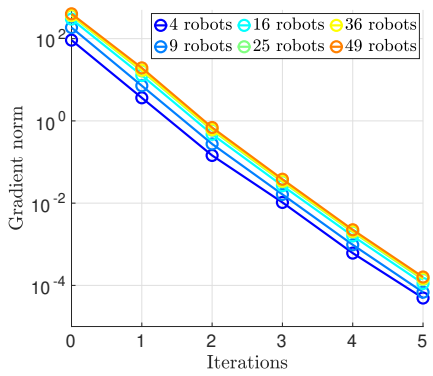
[8] In centralized optimization (*e.g.*, [66, 68–70]), these heuristic sparsifications often serve as preconditioners and need to be used within iterative methods such as conjugate gradient to provide the best performance.



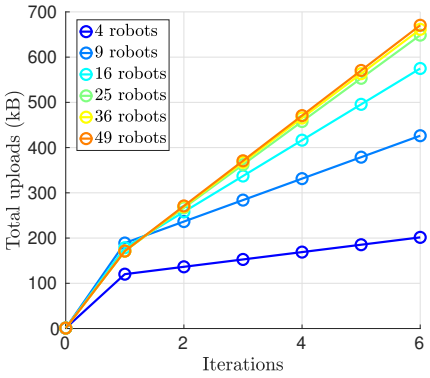
(a) Synthetic rotation averaging



(b) RMSE vs. iterations



(c) Gradient norm vs. iterations

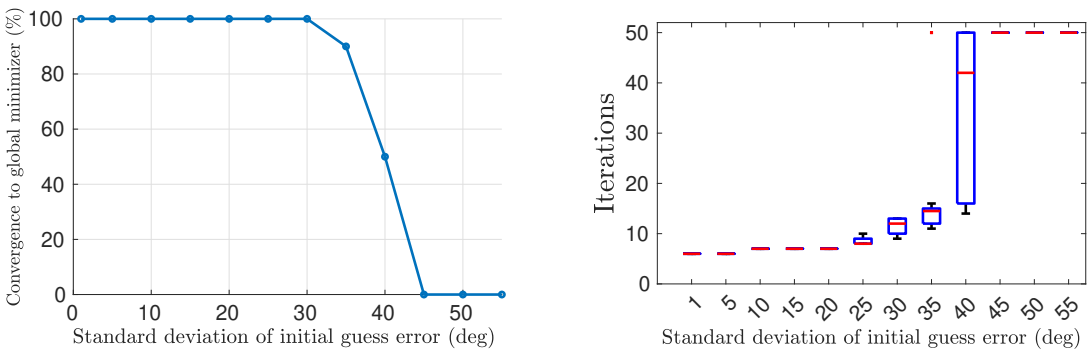


(d) Uploads vs. iterations

Figure 8-6: Scalability of Algorithm 8.4 as the number of robots increases. (a) Synthetic chordal rotation averaging problem with 8000 total rotation variables arranged in a 3D grid. Each edge indicates a relative rotation measurement corrupted by Langevin noise. (b) Evolution of RMSE (in degree) with respect to ground truth rotations. (c) Evolution of Riemannian gradient norm as a function of iterations. (d) Evolution of total uploads as a function of iterations.

in the download stage. Notice that the evolution follows the same trend as Figure 8-5b, where the horizontal axis shows the number of iterations. This observation is expected as a result of Remark 8.6, which shows that the communication complexity in the download stage is $O(mK|\mathcal{C}|)$, *i.e.*, the total downloads grows *linearly* with respect to the number of iterations K.

Scalability with Number of Robots. In this experiment, we evaluate the scalability of Algorithm 8.4. For this purpose, we generate a large-scale synthetic rotation averaging problem with 8000 rotations arranged in a 3D grid (Figure 8-6a). With probability 0.3, we add relative measurements between nearby rotations, which



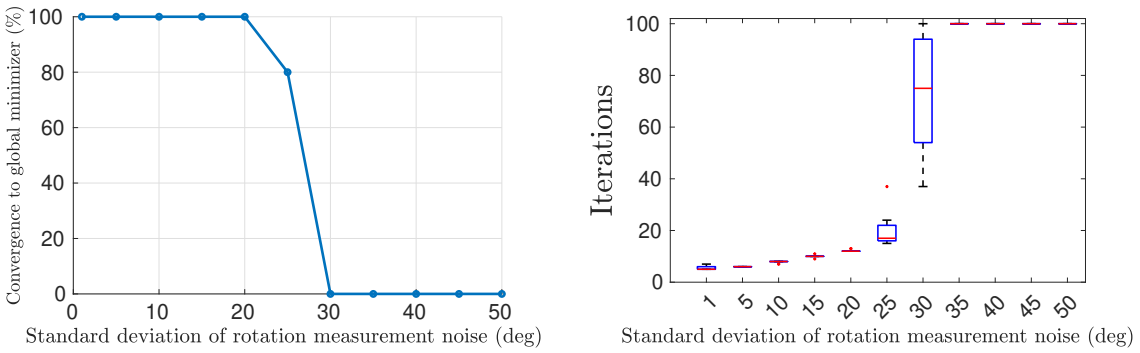
(a) Success rate

(b) Number of iterations

Figure 8-7: Sensitivity of Algorithm 8.4 to accuracy of initial guess. We generate synthetic initial guesses with degrading accuracy by perturbing the global minimizer with increasing levels of Langevin noise. At each level of perturbation, 10 random runs are performed. (a) Percentage of runs that converge to the global minimizer. (b) Boxplot of number of iterations used by Algorithm 8.4.

are corrupted by Langevin noise with a standard deviation of 5 deg. Then, we divide the dataset to simulate increasing number of robots, and run Algorithm 8.4 with sparsification parameter $\epsilon = 0.5$ until the Riemannian gradient norm reaches 10^{-5}. Figure 8-6b 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 8.1 (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. Figure 8-6c shows the evolution of gradient norm as a function of iterations. Note that all curves in Figure 8-6c 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. Lastly, Figure 8-6d 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 $|\mathcal{C}|$ increase, and thus each iteration requires more communication.

Sensitivity to Initial Guess. So far, we have used the distributed chordal initialization technique [38] to initialize Algorithm 8.4. In the next experiment, we test



(a) Success rate

(b) Number of iterations

Figure 8-8: Sensitivity of Algorithm 8.4 to rotation measurement noise. We generate synthetic chordal rotation averaging problems with increasing magnitude of measurement noise. At each noise level, 10 random runs are performed. (a) Percentage of runs that converge to the global minimizer. (b) Boxplot of number of iterations used by Algorithm 8.4.

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 level of Langevin noise. Using the synthetic initialization, we run Algorithm 8.4 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. Figure 8-7a shows the fraction of trials that successfully converge to the global minimizer. We observe that Algorithm 8.4 enjoys a large convergence basin: the success rate only begins to decrease at a large initial guess error of 35 deg. Figure 8-7b shows the number of iterations used by Algorithm 8.4 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 8.4 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. Figure 8-8 shows the results. We find that Algorithm 8.4 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 encountered in practice is usually

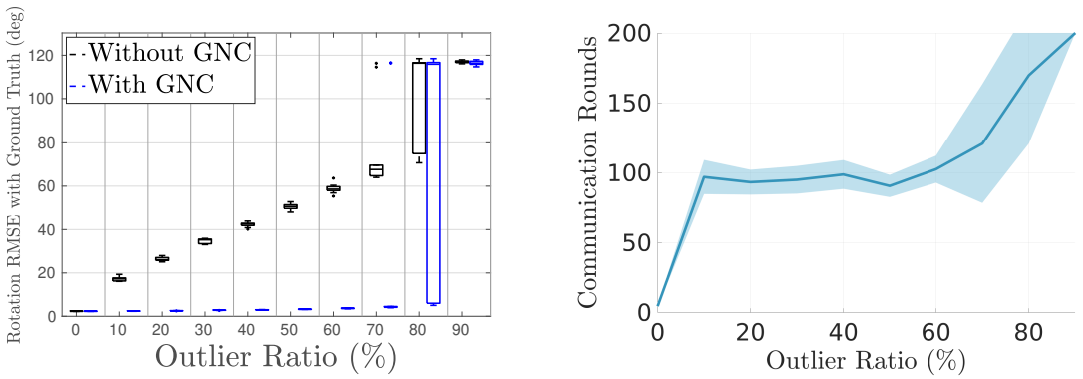
much lower,[9] and thus we expect our algorithm to still provide effective estimation (see real-world evaluations in Sections 8.5.3 and 8.5.4).

Outlier-Robust Optimization. Lastly, we evaluate the proposed outlier-robust optimization method to solve robust rotation averaging problems. In this experiment, we use a 9-robot simulation where each robot owns 512 rotations. In Sections 8.5.3 and 8.5.4, we demonstrate our method on real-world SLAM and SfM problems. As in common SLAM scenarios, we assume each robot has a backbone of odometry measurements within its own trajectory that are free of outliers. Then, with increasing probability, we replace the remaining measurements (corresponding to intra-robot and inter-robot loop closures) with gross outliers. All inlier measurements (including odometry) are corrupted by Langevin noise with a standard deviation of 3 deg, and we set the TLS threshold $\bar{e}$ to correspond to 10 deg. Figure 8-9a visualizes the RMSE with respect to ground truth rotations. As expected, Algorithm 8.4 without GNC is not robust to outliers and shows significant error as soon as outlier measurements are introduced. Nevertheless, by using Algorithm 8.4 within GNC as described in Section 8.4.4, the resulting approach becomes robust and is able to tolerate up to 70% of outlier loop closures. In Figure 8-9b, we study the efficiency of our approach by showing the total number of inner iterations of Algorithm 8.4 used by GNC. Recall that each inner iteration also corresponds to a single round of communication. When the outlier ratio is zero, GNC reduces to the standard Algorithm 8.4 and only requires a few iterations to converge. When outliers are added, GNC requires multiple outer iterations and thus multiple calls to Algorithm 8.4, resulting in increased communication rounds. Nevertheless, for all test cases with less than 70% outlier measurements, the number of communication rounds is approximately 100, which is a reasonable requirement for a real system.

8.5.2 Evaluation on Benchmark PGO Datasets

In this subsection, we evaluate our approach on 12 benchmark pose graph SLAM

[9]Here we only consider rotation noise of *inlier* measurements. *Outlier* measurements will be handled using the robust optimization framework presented in Section 8.4.4.



(a) RMSE with ground truth

(b) Communication rounds

Figure 8-9: Evaluation of robust optimization on synthetic rotation averaging problems corrupted by increasing percentage of outlier loop closures. At each outlier percentage, 10 random runs are performed. (a) RMSE with respect to ground truth rotations. (b) Communication rounds used by GNC. Sold line and shaded area correspond to the mean and one standard deviation, respectively.

datasets. For these datasets, we do not explicitly handle outliers. Outlier-robust estimation will be evaluated in Sections 8.5.3 and 8.5.4.

Evaluation on Rotation Averaging Subproblem. We first evaluate Algorithm 8.4 on the rotation averaging subproblems extracted from the benchmark datasets. For each problem, we simulate a scenario with 5 robots, and run the proposed method (Algorithm 8.4) with sparsification parameter $\epsilon = 1.5$ and the baseline Newton method. Both methods are terminated when the Riemannian gradient norm is smaller than 10^{-5}. Since the spectral sparsification method we use [136] is randomized, we perform 5 random runs of our method. Table 8.1 shows the average number of iterations, uploads, and downloads to reach the desired precision. On all datasets, we are able to verify that all methods converge to the global minima of the considered rotation averaging problems. The proposed method achieves an empirical convergence speed that is close to Newton and typically converges in a few iterations.[10] We note that this is significantly faster than existing fully distributed methods that often require hundreds of iterations to achieve moderate precision. For

[10]One notable exception is the Rim dataset, for which our method uses more than 20 iterations to converge. A closer investigation reveals that this dataset actually contains some outlier measurements. Specifically, at the global minimizer $R^\star$, there are 28 measurements $\widetilde{R}_{ij}$ for which $\mathbf{d}(R_i^\star \widetilde{R}_{ij}, R_j^\star) > 60$ deg. Since these outliers have large residuals, their contributions to the Hessian can no longer be well approximated by the corresponding Laplacian terms. As a result, the performance of our method is negatively impacted.

Table 8.1: Rotation averaging on benchmark SLAM datasets with 5 robots. $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the total number of rotation variables and measurements, respectively. We run the baseline Newton method and the proposed method (Algorithm 8.4) with sparsification parameter $\epsilon = 1.5$, and compare the number of iterations, uploads, and downloads to reach a Riemannian gradient norm of 10^{-5}. For the proposed method, we also show the sparsity achieved by sparsification (lower is better). Results averaged across 5 runs.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Iterations		Upload (kB)		Download (kB)		Achieved sparsity
			Newton	Proposed	Newton	Proposed	Newton	Proposed	by proposed (%)
Killian Court (2D)	808	827	2	3	1.6	1.1	0.5	0.8	100
CSAIL (2D)	1045	1171	2	4	7.2	5.9	2.3	4.6	97.3
INTEL (2D)	1228	1483	3	4.2	10.5	5.8	3.3	4.6	96.4
Manhattan (2D)	3500	5453	2	5	118.9	49.5	12.5	31.3	38.7
KITTI 00 (2D)	4541	4676	2	2	13.2	6.6	4.4	4.4	100
City (2D)	10000	20687	2	4	450.3	351.5	129	258.1	97.3
Garage (3D)	1661	6275	1	2	274.4	88.9	35.8	71.6	93.2
Sphere (3D)	2500	4949	2	8.6	2548.8	106	19.2	82.6	16.9
Torus (3D)	5000	9048	3	9.6	10423.7	229.5	57	182.2	12.4
Grid (3D)	8000	22236	3	9.2	206871.6	886.6	220.8	677	2.7
Cubicle (3D)	5750	16869	2	6.8	7015	440.3	107.7	366.2	19.9
Rim (3D)	10195	29743	4	23.4	53657.9	1320.9	209.1	1223.2	6.6

both Newton and the proposed method, the total download is proportional to the number of iterations (see Remark 8.6). Thus, our method uses more downloads since it requires more iterations. However, we note that compared to the download stage, the upload stage is more communication-intensive since robots need to transmit (potentially dense) Schur complements to the server. Using spectral sparsification, the proposed approach achieves significant reduction in uploads, especially on challenging datasets such as Grid and Rim. Finally, the last column of Table 8.1 shows the achieved sparsification as the percentage of nonzero elements that remain after spectral sparsification. We observe that the benefit of sparsification varies across datasets. For example, on Killian Court and INTEL, the effect of sparsification is limited because the exact Schur complement S is already sparse. Meanwhile, on datasets such as Grid and Rim, the benefit of sparsification is substantial and the results have less than 10% nonzero elements. In Section 8.5.5, we provide a thorough discussion on the impact of problem properties on sparsification performance.

Sparsification Runtime. Recall that in the SparsifiedSchurComplement step in Algorithm 8.4, each robot α sparsifies its S_α matrix and transmits the result $\widetilde{S}_\alpha$ to the server. This step uses the majority of robots' local computation time. In Figure 8-10, we evaluate the runtime of the sparsification algorithm [136] on the 12

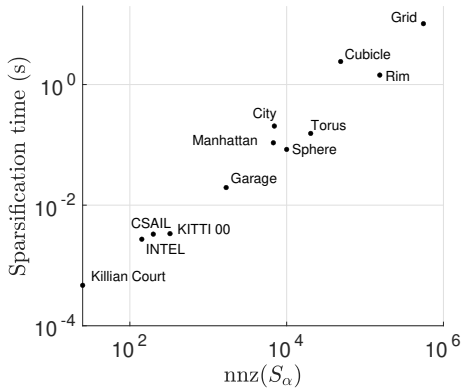


Figure 8-10: Spectral sparsification runtime on benchmark datasets.

Table 8.2: PGO initialization on benchmark SLAM datasets with 5 robots. e^{Σ} measures the average error in marginal covariance due to decoupled rotation and translation estimation. Optimality gap and RMSE are computed with respect to optimal solutions from SE-Sync [14]. In addition, we also show the number of iterations and communication (both upload and download) used by the rotation and translation estimation stages in our approach, and compare the results with RBCD++ (Chapter 4) to achieve the same optimality gap.

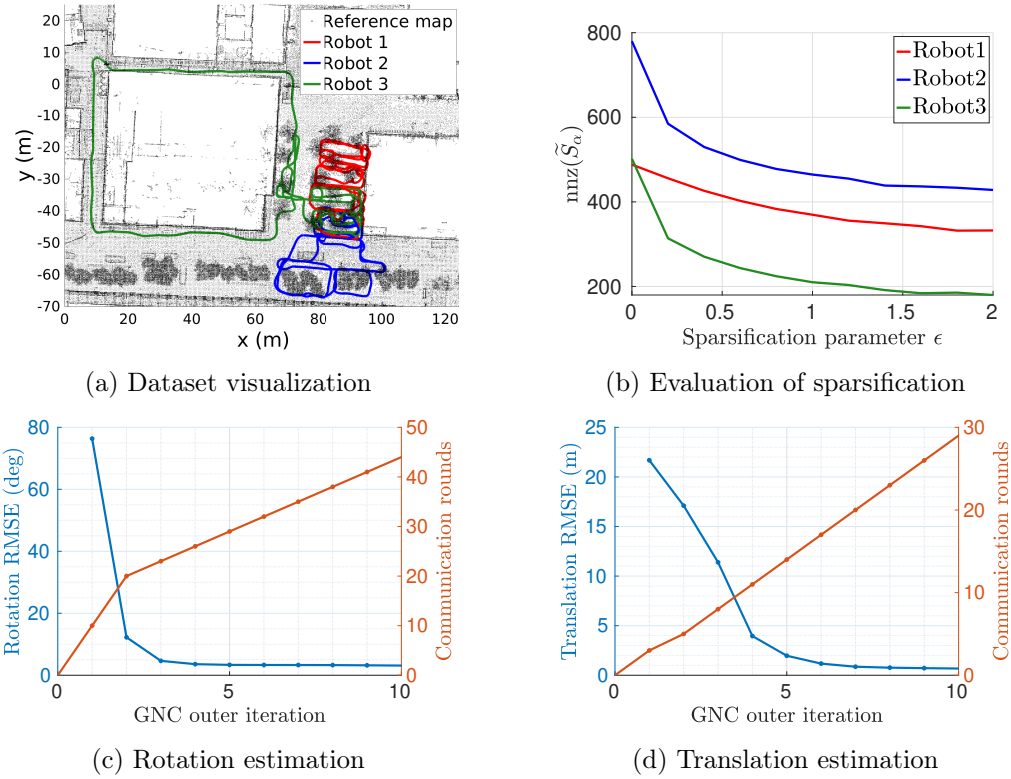
Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	e^{Σ}	Optimality Gap	RMSE with optimal PGO solution		Iterations			Total communication (kB)		
					Rotation (deg)	Translation (m)	Rot.	Tran.	RBCD++	Rot.	Tran.	RBCD++
Killian Court (2D)	808	827	0.76	0.12	4.48	4.12	5	2	173	3.0	2.4	246
CSAIL (2D)	1045	1171	0.33	4.6×10^{-4}	0.06	0.01	3	3	442	8.3	15.2	1.9×10^3
INTEL (2D)	1228	1483	0.21	2.2×10^{-3}	0.36	0.03	4	4	68	10.0	18.7	370
Manhattan (2D)	3500	5453	0.92	0.15	1.75	0.47	4	5	57	72.8	147.8	1.3×10^3
KITTI 00 (2D)	4541	4676	0.86	0.33	0.46	0.64	3	2	289	15.4	19.8	3.1×10^3
City (2D)	10000	20687	0.95	0.12	0.63	0.18	4	4	1000	611	1.1×10^3	1.9×10^5
Garage (3D)	1661	6275	0.99	0.12	0.43	0.33	2	2	59	161	162	3.7×10^3
Sphere (3D)	2500	4949	0.87	0.17	1.39	0.38	7	7	89	185	185	2.9×10^3
Torus (3D)	5000	9048	0.25	0.01	2.15	0.07	8	6	113	394	317	4.8×10^3
Grid (3D)	8000	22236	0.43	0.03	1.22	0.06	8	8	102	1.7×10^3	1.7×10^3	2.5×10^4
Cubicle (3D)	5750	16869	0.86	0.18	1.53	0.16	7	6	42	869	722	4.4×10^3
Rim (3D)	10195	29743	0.79	0.63	4.95	0.78	25	6	102	2.8×10^3	748	6.8×10^3

benchmark datasets shown in Table 8.1. For each dataset, we record the maximum sparsification time among all robots, and visualize the result as a function of the number of nonzero entries in the input matrix S_α. On most datasets, the maximum runtime is below one second. On the Grid dataset, the input matrix has more than 5×10^5 nonzero entries and our implementation uses 10.2 seconds. Overall, we conclude that the runtime of our implementation is still reasonable. However, we believe that further improvements are possible, *e.g.*, by approximately computing effective resistances during spectral sparsification as suggested in [136].

Initialization for PGO. Lastly, we evaluate the use of our methods to initialize PGO. Recall from Section 8.2 that our initialization scheme involves two stages.

First, we initialize rotations by solving the rotation averaging subproblem in PGO. Then, fixing the rotation estimates in the PGO cost function (8.8), we can initialize translations by solving the resulting translation estimation subproblem. For this experiment, we start Algorithm 8.4 at an initial guess computed from a spanning tree of the pose graph. This is done to demonstrate that our method does not need to rely on distributed chordal initialization [38], which is itself an iterative procedure. Table 8.2 reports the optimality gap and estimation RMSE of our initialization method compared to the optimal PGO solutions computed using SE-Sync [14]. Our results show that the quality of initialization varies across datasets. In general, since our initialization method decouples the estimation of rotations and translations, we expect its performance to degrade when there is significant coupling between rotation and translation terms in the full PGO problem. To investigate this hypothesis, we treat PGO as an inference problem over factor graphs [19] and consider the covariance Σ^{PGO} of the pose estimates at the optimal solution. We compare Σ^{PGO} with the corresponding covariance Σ^{INIT} produced by our two-stage initialization, where the rotation and translation blocks of Σ^{INIT} are extracted from rotation averaging and translation estimation, respectively. Since both covariance matrices are large and dense, we only compute their diagonal blocks Σ_i^{PGO} and Σ_i^{INIT} that correspond to the marginal covariances of pose i. We quantify the error introduced by decoupled rotation and translation estimation by computing the normalized error $e_i^{\Sigma} = \|\Sigma_i^{\text{PGO}} - \Sigma_i^{\text{INIT}}\|_F \,/\, \|\Sigma_i^{\text{PGO}}\|_F$. Table 8.2 reports e^{Σ}, which is the average of e_i^{Σ} over all poses. We find that the results separate all datasets into two groups. INTEL, CSAIL, Torus, and Grid have small values of e^{Σ}, and our initialization achieves the best performance, especially in terms of optimality gap. On the remaining datasets with larger values of e^{Σ}, the two-stage initialization produces worse results. Lastly, we note that Rim is a special case due to the presence of outlier measurements.

In addition, Table 8.2 also reports the number of iterations and total communication (both uploads and downloads) used by our initialization during rotation estimation and translation estimation. To provide additional context, we also report corresponding results for the RBCD++ solver developed in Chapter 4 to achieve the



(a) Dataset visualization

(b) Evaluation of sparsification

(c) Rotation estimation

(d) Translation estimation

Figure 8-11: Robust PGO initialization on real-world collaborative SLAM dataset. (a) Trajectory estimates produced by the proposed robust PGO initialization, which are qualitatively overlaid on top of a point cloud map of the experiment area. The point cloud map was created at an earlier time (certain objects such as cars have changed) and is included only for visualization. (b) Evaluation of spectral sparsification. (c) Rotation RMSE as a function of communication rounds during rotation estimation. (d) Translation RMSE as a function of communication rounds during translation estimation.

same optimality gap from the same initial guess. We note that the RBCD++ results are only included for reference since this method is fully distributed whereas our method assumes a server-client architecture. Furthermore, given more iterations, RBCD++ will eventually achieve better accuracy because the method is solving the full PGO problem. However, our results still suggest that when a server-client architecture is available, our method is favorable and provides high-quality initialization using only a few iterations.

8.5.3 Robust PGO Initialization for Real-World CSLAM

In this section, we show that our approach can be used to achieve *robust* PGO ini-

tialization in a real-world collaborative SLAM (CSLAM) 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 (Figure 8-11a). We run Kimera-Multi (Chapter 6) to process the dataset as a 3-robot collaborative visual SLAM mission. The resulting multi-robot pose graph contains a 3D 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.

We demonstrate the two-stage PGO initialization as described in Section 8.2. 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%). Table 8.3 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 [6]. 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. Figure 8-11b evaluates the effects of spectral sparsification on the real-world dataset. We observe similar benefits as in previous experiments, where enabling sparsification ($\epsilon > 0$) significantly reduces the number of nonzero entries each robot needs to communicate. Lastly, Figures 8-11c and 8-11d evaluate the efficiency of our two-stage robust initialization, by visualizing the evolution of RMSE and number of communication rounds as a function of GNC outer iterations. Recall that each communication round also corresponds to a single

Table 8.3: Evaluation of robust PGO initialization on real-world collaborative SLAM dataset.

Robot	Length (m)	Keyframes	RMSE without GNC		RMSE with GNC	
			Rot. (deg)	Tran. (m)	Rot. (deg)	Tran. (m)
1	483	3192	67.5	11.9	1.8	0.5
2	458	2518	73.1	17.6	3.3	0.7
3	524	3374	86.8	23.6	3.7	0.6

iteration of Algorithm 8.4 or Algorithm 8.5. For this experiment, the sparsification parameter is fixed at $\epsilon = 2$. Overall, for both rotation estimation and translation estimation, the RMSE converges after a few GNC outer iterations, and consequently, only a small number of communication rounds is needed.

Evaluations on large-scale CSLAM datasets. To conclude this subsection, we extend our previous evaluations to large-scale CSLAM datasets. To this end, we use the lidar-centric Nebula multi-robot datasets [3] and the vision-based Kimera-Multi datasets presented in Section 6.7. Tables 8.4 and 8.5 report the performance of the proposed two-stage PGO initialization scheme on these datasets. We compare our two-stage approach against the initial guesses (computed using the robust initialization method in Section 6.4.2) as well as reference solutions obtained using the GNC implementation in GTSAM [6]. On all datasets, the two-stage approach improves over the initial guesses and furthermore achieves estimation accuracy that is comparable to or slightly worse than the GTSAM solutions. For the proposed method, we also report additional information including the average achieved sparsity by spectral sparsification, number of iterations (communication rounds), total communication (both uploads and downloads), and runtimes on a single computer.[11] We note that spectral sparsification achieves limited sparsity improvements on the Kimera-Multi datasets in Table 8.5. This is because on these datasets, the exact Schur complements are already sparse, and thus sparsification is not necessary; see Section 8.5.5 for an in-depth discussion.

[11]The runtime results are obtained by running the proposed method on a single computer and thus do not take into account the effect of communication delays. Future work will improve the runtime evaluation by implementing the proposed approach on real-world multi-robot systems.

Table 8.4: Evaluation of robust PGO initialization on the Nebula multi-robot datasets [3].

Datasets	Distance (m)	Robots	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	RMSE against ground truth (m)			Additional Information			
					Initial	Two-Stage	GTSAM	Achieved sparsity (%)	Iterations	Comm. (MB)	Runtime (sec)
Tunnel	2556	2	1278	6832	1.68	1.13	1.06	47.9	112	5.62	8.1
Urban	1530	3	765	2214	1.27	1.11	1.23	63.1	93	1.33	1.8
Prelim2	1224	4	613	2072	0.88	0.29	0.38	83.3	104	2.29	1.9
KU	6304	4	3153	4034	5.56	1.75	1.18	94.8	88	0.85	3.4

Table 8.5: Evaluation of robust PGO initialization on the Kimera-Multi field experiment datasets (Section 6.7).

Datasets	Distance (m)	Robots	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	RMSE against ground truth (m)			Additional Information			
					Initial	Two-Stage	GTSAM	Achieved sparsity (%)	Iterations	Comm. (MB)	Runtime (sec)
Campus-Outdoor	6044	6	4015	4196	27.76	13.92	11.61	100	49	0.57	3.6
Campus-Tunnels	6753	8	4952	13231	22.67	4.01	4.03	98.8	76	11.88	10.8
Campus-Hybrid	7785	8	5502	6592	29.29	6.77	6.84	98.6	68	3.72	6.1

8.5.4 Evaluation on Real-World SfM Datasets

Lastly, we evaluate our method on rotation averaging problems extracted from 15 real-world structure-from-motion (SfM) datasets [206]. Each dataset is a collection of many internet images taken at a particular location. We use Theia [204] 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 [204] and use 3D reconstructions produced by the incremental SfM pipeline [207] as reference solutions. Table 8.6 reports full dataset statistics.

We divide each dataset to simulate 5 robots and run our GNC-based rotation averaging solver, with the TLS threshold set to 5 deg. Unlike collaborative SLAM, each robot no longer has an outlier-free odometry backbone in SfM. This also means that we cannot use the approach in Section 6.4 to compute an outlier-free initial guess for the variable update step in GNC (see Remark 8.7). Instead, we use the initial guess from Theia that is computed using a spanning tree of the measurement graph. Table 8.6 reports the mean estimation error. On 14 out of the 15 datasets, our GNC-based robust rotation averaging produces accurate results that significantly improve from the initial guesses. The only failure case, Gendarmenmarkt, is known to be a very challenging case in which the underlying 3D scene is highly symmetric, consisting of two visually similar churches. The symmetry leads to a significantly lower percentage of good inlier measurements and also causes the spanning tree initial guess to have a

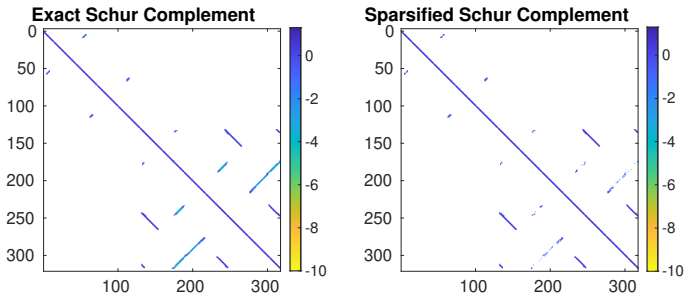
Table 8.6: Robust rotation averaging on real-world SfM datasets. Each dataset is divided to simulate 5 robots. $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the total number of rotation variables and measurements, respectively. Using the reference solution, we quantify the difficulty of each dataset by 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.

DATASETS	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Measurement Quality (%)			Mean Error (deg)			Achieved	Communication (MB)	
			Inlier	Outlier	Other	Initial	No GNC	With GNC	sparsity (%)	Upload	Download
Montreal Notre Dame	468	49705	81	4	15	4.2	3.3	1.1	55.1	2.09	1.21
Ellis Island	241	19507	63	1	26	7.0	5.6	2.3	66.0	0.95	0.57
NYC Library	355	17579	61	6	33	4.3	4.3	2.3	75.8	1.24	0.83
Notre Dame	553	97764	70	9	21	3.5	4.5	2.4	41.8	2.85	1.54
Roman Forum	1099	53989	74	3	23	16.5	5.1	2.5	68.1	4.43	2.73
Alamo	606	87725	74	3	23	8.0	4.5	2.9	44.7	2.62	1.39
Madrid Metropolis	379	18811	47	20	33	7.7	8.0	3.4	70.3	1.48	1.08
Yorkminster	448	24416	73	5	22	8.3	4.5	3.4	76.5	1.69	1.1
Tower of London	493	19798	76	3	21	7.4	4.7	3.5	75.3	2.00	1.35
Trafalgar	5433	680012	63	7	30	20.0	6.4	3.5	40.7	35.89	24.97
Piazza del Popolo	343	22342	82	4	14	5.4	7.8	3.6	70.0	1.29	0.8
Piccadilly	2436	254175	58	10	32	13.9	14.6	4.9	53.1	14.23	9.79
Union Square	930	25561	57	6	37	11.9	10.9	6.0	82.1	4.15	3.37
Vienna Cathedral	900	96546	70	6	24	13.9	9.6	8.9	51.9	4.62	3.13
Gendarmenmarkt	723	42980	36	27	37	45.0	40.8	38.1	63.8	4.13	3.03

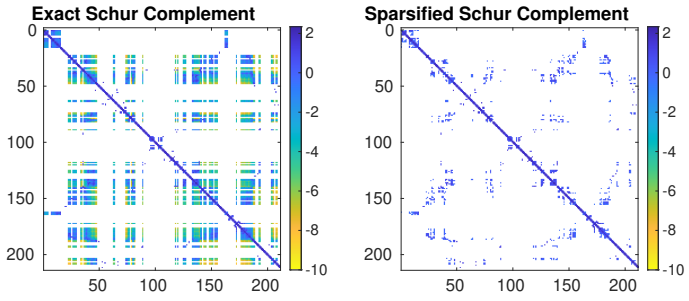
large error, which GNC is unable to recover from.[12] In summary, we conclude that on most datasets, our proposed rotation averaging solver combined with GNC is able to achieve robust rotation estimation, despite outlier measurements and the increased noise level present in internet images.

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 8.5.5, we explain the reasons behind using the increased value for ϵ. Table 8.6 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.

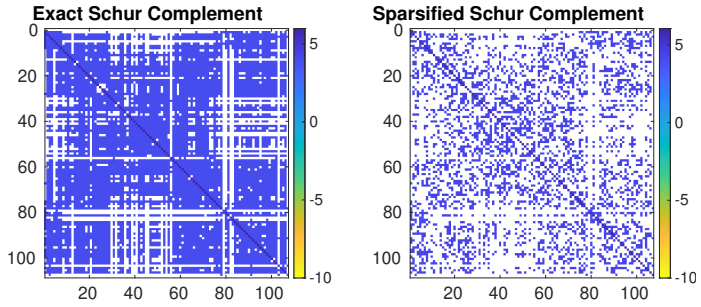
[12] We note that our result on Gendarmenmarkt is also consistent with performance reported by Theia, which shows a large final error: http://theia-sfm.org/performance.html.



(a) Garage ($\epsilon = 1.5$)



(b) Manhattan ($\epsilon = 1.5$)



(c) Notre Dame ($\epsilon = 5.0$)

Figure 8-12: 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_α in Algorithm 8.2), and the result after spectral sparsification (corresponding to $\widetilde{S}_\alpha$ in Algorithm 8.2). Entries in the matrix are color-coded based on their magnitude in log scale.

8.5.5 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 (Theorem 8.1). When there are outlier measurements, we have seen that the approximation quality de-

grades, leading to increased number of iterations. An example is the Rim dataset in Table 8.1, 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 Figure 8-11c. During the first two GNC outer iterations, outliers have a substantial influence on the problem, causing our method (Algorithm 8.4) to use more communication rounds. However, as GNC proceeds, outliers receive increasingly small weights, and our method recovers its fast convergence. In Figure 8-11c, 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 8.1), 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 8.2). 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 Figure 8-12 shows a representative sparsification result for each case. In the first case (Figure 8-12a), the multi-robot 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 Figure 8-12b, 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 ϵ. In the last case, the original measurement graph is already dense and so are the resulting Schur complements (Figure 8-12c). All of the SfM datasets in Table 8.6 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 ϵ is needed. Similar to the second case, sparsification is highly effective at promoting sparsity in each robot's transmitted matrix in this case.

8.6 Conclusion

This chapter presented fast and communication-efficient methods for solving rotation averaging and translation estimation in multi-robot 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.

Chapter 9

Conclusion

This thesis has developed optimization algorithms and multi-robot systems toward scalable collaborative geometric estimation. The first part of the thesis focuses on the fully distributed communication architecture. Chapter 4 presented DC2-PGO, a distributed certifiably correct PGO solver based on a sparse semidefinite relaxation. In Chapter 5, we further adapt the distributed local optimization algorithm to operate under asynchronous communication, and showed that the resulting algorithm, ASAPP, maintains first-order convergence guarantees under bounded delay. Utilizing the developed distributed optimization approach, Chapter 6 developed Kimera-Multi, an outlier-robust and fully distributed system for metric-semantic CSLAM, and presented extensive evaluations in both photo-realistic simulations and large-scale field experiments involving up to 8 robots. The second part of the thesis focuses on the server-client communication architecture. Chapter 7 presented LARPG, a communication-efficient collaborative solver for large-scale bundle adjustment based on event-triggered communication. Lastly, Chapter 8 presented specialized collaborative solvers for collaborative rotation averaging, translation estimation, and two-stage pose graph initialization that achieve fast convergence by leveraging fast Laplacian solvers and spectral sparsification.

9.1 Future Work

While the results presented in this thesis are promising, they also suggest the following interesting future directions.

Hierarchical Optimization and Communication. While the distributed optimization algorithms developed in Chapters 4 and 5 are flexible and faster than prior fully distributed methods, they still could exhibit slow convergence on very large-scale problem instances, *e.g.*, the ones encountered in the Kimera-Multi field experiments (Section 6.7). An interesting direction to address this challenge is *hierarchical* optimization, which aims to tackle the optimization problem at multiple levels of resolution. Intuitively, performing optimization at coarser levels helps to more efficiently correct *long-range* or *low-frequency errors* that are otherwise hard to eliminate at the fine level. This idea lies at the heart of multigrid methods in fast linear solvers (*e.g.*, [67, Chapter 13]), and has also been explored in prior works in single-agent SLAM (*e.g.*, [150, 208, 209]). Nevertheless, hierarchical optimization has been less explored in the multi-agent domain. To this end, leveraging the techniques developed in this thesis under a hierarchical or multi-resolution formulation could be an interesting direction to investigate.

In addition to hierarchical optimization, *hierarchical communication* is another interesting avenue to explore. In our applications, the distinction between the two notions is that the former defines hierarchy on the *measurement graph* (*e.g.*, the multi-robot pose graph), while the latter considers a hierarchical *communication graph*. From this perspective, the server-client architecture (Figure 1-1b) can be viewed as a hierarchical communication graph with two levels, and it would be interesting to explore the generalization of the algorithms developed in this thesis (*e.g.*, the use of spectral sparsification in Chapter 8) to hierarchical communication graphs with multiple levels. Lastly, studying the interplay between the measurement graph and the communication graph is another interesting open question.

Collaborative Estimation of More Expressive Models. This thesis considers collaborative estimation problems that involve standard geometric models, *e.g.*,

pose graphs for collaborative localization and point landmarks for collaborative mapping (bundle adjustment). Meanwhile, state-of-the-art single-agent perception systems are able to construct increasingly expressive models of the environment that facilitate higher-level inference and planning. It would be interesting to extend the methods developed in this thesis to support these more expressive models. An example is to generalize distributed optimization algorithms to support maps with *learned* representations (*e.g.*, learned embeddings that parametrize object shapes [210]). As a second example, recent works have moved beyond the standard SLAM formulation by constructing hierarchical maps that unify geometric, semantic, and topological representations [36, 200, 211]. While there are preliminary works that generalize these new representations to the multi-agent domain [212, 213], robustly and efficiently solving the resulting large-scale optimization problems still remains an open challenge.

Improving and Generalizing Theoretical Guarantees. This thesis has established performance guarantees (*e.g.*, in terms of convergence) for the developed collaborative optimization algorithms. Nevertheless, the theoretical guarantees often appear conservative compared to what was observed in numerical experiments, and it would be useful to improve the theoretical analysis to bridge this gap. For example, in Chapter 7, it would be useful to relax the assumptions needed to establish the convergence guarantees. In Chapter 8, it would be highly useful to generalize the Hessian approximation result in Theorem 8.1 to problems such as PGO and to derive explicit bound for the approximation constant δ. For the latter, considering the statistics of the measurement model (instead of working with realizations of measurement errors) could be useful. In addition, it would be interesting to investigate whether the convergence rate established in Theorem 8.3 is optimal from the perspective of iteration complexity.

Appendix A

Supplemental Materials for Chapter 4

A.1 Exactness of SDP Relaxation

In this section, we provide additional discussions of our SDP relaxation (Problem 4.1) and give a proof of its exactness under low noise. To facilitate the discussion, we summarize all problems that are considered in this work in Table A.1. The core idea behind our proof is to establish certain *equivalence* relations with the rotation-only SDP relaxation (Problem 4.2), which is first developed by Rosen *et al.* in SE-Sync [14]. To begin, we first define the cost matrix Q_{R} that appears in the rotation-only SDP; see also [14, Equation 20(b)].

$$Q_{\mathrm{R}} \triangleq L_{\mathrm{R}} + \widetilde{\Sigma} - \widetilde{V}^{\top} L(W^{\tau})^{\dagger} \widetilde{V}, \tag{A.1}$$

In (A.1), $L_{\mathrm{R}} \in \mathcal{S}_{+}^{dn}$ is the rotation connection Laplacian, and $L(W^{\tau}) \in \mathcal{S}_{+}^{n}$ is the graph Laplacian of the pose graph with edges weighted by the translation measurement weights $\{\tau_{ij}\}$. The remaining two matrices $\widetilde{V} \in \mathbb{R}^{n \times dn}$ and $\widetilde{\Sigma} \in \mathbb{R}^{dn \times dn}$ are formed using relative translation measurements. The exact expressions of these matrices are given in equations (13)-(16) in [14] and are omitted here. Let us consider the rank-restricted version of the rotation-only SDP defined below.

Table A.1: List of problems considered in Chapter 4. Here $i \in [n]$ where n denotes the total number of poses in the (collective) pose graph. The dimension of the problem in denoted by $d \in \{2, 3\}$. We note that Problem 4.2 and A.1 are not directly used in the proposed approach, but are nonetheless crucial for establishing the performance guarantees of the SDP relaxation for pose synchronization (Theorem 4.1 and Theorem 4.2).

#	**Problem Description**	**Cost Function**	**Domain**	**Constraints**
2.2	MLE for PGO	(2.20)	$(R_i, t_i) \in \mathrm{SO}(d) \times \mathbb{R}^d$	–
4.1	Full SDP Relaxation of PGO	$F(Z) \triangleq \langle Q, Z \rangle$	$Z \in \mathcal{S}_+^{n+dn}$	$Z_{[i,i](1:d,1:d)} = I_d$
4.2	Rotation-only SDP Relaxation for PGO	$F_{\mathrm{R}}(Z_{\mathrm{R}}) \triangleq \langle Q_{\mathrm{R}}, Z_{\mathrm{R}} \rangle$	$Z_{\mathrm{R}} \in \mathcal{S}_+^{dn}$	$Z_{\mathrm{R}[i,i]} = I_d$
4.3	Rank-Restricted Full SDP for PGO	$f(X) \triangleq \langle Q, X^\top X \rangle$	$X \in (\mathrm{St}(d, r) \times \mathbb{R}^r)^n$	–
A.1	Rank-Restricted Rotation-only SDP for PGO	$f_{\mathrm{R}}(Y) \triangleq \langle Q_{\mathrm{R}}, Y^\top Y \rangle$	$Y \in \mathrm{St}(d, r)^n$	–

Problem A.1 (Rotation-only Rank-restricted SDP for Pose Synchronization [14])**.**

$$\underset{Y \in \mathrm{St}(d,r)^n}{\text{minimize}} \quad \langle Q_{\mathrm{R}}, Y^\top Y \rangle. \tag{A.2}$$

Problem A.1 and the *sparse* rank-restricted relaxation we solve in this work (Problem 4.3) are intimately connected. The following lemma precisely characterizes this connection, and also provides an important tool for proving subsequent theorems in this section.

Lemma A.1 (Connections between Problems 4.3 and A.1)**.** *Let $X = [Y_1 \; p_1 \; \dots \; Y_n \; p_n] \in \mathcal{M}_{\mathrm{PGO}}(r, n)$ be a first-order critical point of Problem 4.3. Let $Y = [Y_1 \; \dots \; Y_n] \in \mathrm{St}(d, r)^n$ and $p = [p_1 \; \dots \; p_n] \in \mathbb{R}^{r \times n}$ be block matrices constructed from the Stiefel and Euclidean elements of X, respectively. Then:*

(i) The translations $p \in \mathbb{R}^{r \times n}$ satisfy:

$$p \in \left\{ -Y\widetilde{V}^\top L(W^\tau)^\dagger + c1_n^\top \middle| c \in \mathbb{R}^r \right\}. \tag{A.3}$$

(ii) Y is a first-order critical point of Problem A.1, and $\langle Q, X^\top X \rangle = \langle Q_R, Y^\top Y \rangle$.

(iii) Let $\bar{\Lambda}(X)$ denote the symmetric $(d \times d)$-block-diagonal matrix constructed by extracting the nonzero $(d \times d)$ diagonal blocks from the Lagrange multiplier matrix $\Lambda(X)$ defined in (4.43a)*:*

$$\bar{\Lambda}(X) \in \mathrm{SBD}(d, n) \tag{A.4}$$

$$\bar{\Lambda}(X)_{[i,i]} \triangleq \Lambda(X)_{[i,i](1:d,1:d)} \quad \forall i \in [n] \tag{A.5}$$

(see also (4.3)*). Then the Lagrange multiplier matrix:*

$$\Lambda_R(Y) \triangleq \mathrm{SymBlockDiag}(Q_R Y^\top Y) \tag{A.6}$$

for the simplified (rotation-only) Problem A.1 (cf. [14, eq. (107)]) satisfies:

$$\Lambda_R(Y) = \bar{\Lambda}(X). \tag{A.7}$$

(iv) Let $S_R(Y) \triangleq Q_R - \Lambda_R(Y)$ denote the certificate matrix for the simplified (rotation-only) SE-Sync relaxation Problem A.1 [14, Thm. 7]. Then $S_R(Y) \succeq 0$ if and only if $S(X) \succeq 0$.

(v) X is a global minimizer of Problem 4.3 if and only if Y is a global minimizer to Problem A.1.

Proof. Using the definition of the connection Laplacian matrix Q in Problem 4.3 (see [20, Appendix II]), it can be shown that the cost function in Problem 4.3 can be expanded into the following,

$$\langle Q, X^\top X \rangle = \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \left\| Y_j - Y_i \widetilde{R}_{ij} \right\|_F^2 + \sum_{(i,j)\in\mathcal{E}} \tau_{ij} \left\| p_j - p_i - Y_i \widetilde{t}_{ij} \right\|_2^2. \tag{A.8}$$

Using (A.8), we may rewrite the cost function in a way that separates the Euclidean

(translation) variables from the Stiefel (rotation) variables:

$$\langle Q, X^\top X\rangle = \begin{bmatrix}\mathrm{vec}(p)\\ \mathrm{vec}(Y)\end{bmatrix}^\top (M \otimes I_r) \begin{bmatrix}\mathrm{vec}(p)\\ \mathrm{vec}(Y)\end{bmatrix} = \mathrm{tr}\left(\begin{bmatrix}p & Y\end{bmatrix} M \begin{bmatrix}p & Y\end{bmatrix}^\top\right). \tag{A.9}$$

$$M \triangleq \begin{bmatrix} L(W^\tau) & \widetilde{V} \\ \widetilde{V}^\top & L_{\mathrm{R}} + \widetilde{\Sigma}\end{bmatrix}. \tag{A.10}$$

Above, the vec$(\cdot)$ operator concatenates columns of the input matrix into a single vector. A detailed derivation for (A.9) is already presented in [14, Appendix B] for the case when $r = d$. For $r \geq d$, the derivation is largely identical with minor modifications to the dimensions of certain matrices, and thus is omitted. We make an additional remark that the new data matrix M (A.10) is related to the original connection Laplacian Q via a permutation of the columns and rows.

Let us define $f(X) \triangleq \langle Q, X^\top X\rangle$ and $f_{\mathrm{R}}(Y) \triangleq \langle Q_{\mathrm{R}}, Y^\top Y\rangle$. From (A.9), we derive the Euclidean gradients of $f(X)$ with respect to p and Y, respectively.

$$\nabla_p f(X) = 2\left(pL(W^\tau) + Y\widetilde{V}^\top\right), \tag{A.11}$$

$$\nabla_Y f(X) = 2\left(p\widetilde{V} + Y(L_{\mathrm{R}} + \widetilde{\Sigma})\right). \tag{A.12}$$

Similarly, we also have:

$$\nabla_Y f_{\mathrm{R}}(Y) = 2\, Y Q_{\mathrm{R}}. \tag{A.13}$$

Part (i): Since X is a first-order critical point, the Euclidean gradient with respect to the translations must be zero. In light of (A.11), we need to identify p such that,

$$pL(W^\tau) + Y\widetilde{V}^\top = 0. \tag{A.14}$$

Using the general fact that $\mathrm{vec}(ABC) = (C^\top \otimes A)\mathrm{vec}(B)$, we may vectorize the above system of equations to,

$$(L(W^\tau) \otimes I_r)\mathrm{vec}(p) + \mathrm{vec}(Y\widetilde{V}^\top) = 0. \tag{A.15}$$

Define $A \triangleq L(W^\tau) \otimes I_r$ and $b \triangleq \mathrm{vec}(Y\widetilde{V}^\top)$. Since A is the Kronecker product between the Laplacian of a connected graph and the identity matrix, it holds that $\mathrm{rank}(A) = rn - r$ and the kernel of A is spanned by the columns of $U \triangleq 1_n \otimes I_r$. We can equivalently express $\ker(A)$ as (cf. [14, Equation (76)]),

$$\ker(A) = \{Uc \,|\, c \in \mathbb{R}^r\} = \{\mathrm{vec}(c1_n^\top) \,|\, c \in \mathbb{R}^r\}. \tag{A.16}$$

Let $u = \mathrm{vec}(c1_n^\top) \in \ker(A)$ be an arbitrary null vector of A. Consider the inner product between u and b,

$$\langle b, u\rangle = \langle Y\widetilde{V}^\top, c1_n^\top\rangle = \mathrm{tr}(1_n^\top \widetilde{V} Y^\top c) = 0. \tag{A.17}$$

The last equality is due to the fact that $1_n^\top \widetilde{V} = 0$ by the definition of $\widetilde{V}$; see equation (15) in [14]. Therefore, we have proven that $b \perp \ker(A)$. Since A is symmetric positive-semidefinite, it holds that $b \in \mathrm{range}(A)$. Thus, the system of linear equations (A.15) admits infinitely many solutions, characterized by the following set,

$$\mathrm{vec}(p) \in \left\{ -A^\dagger b + Uc \,\middle|\, c \in \mathbb{R}^r \right\}. \tag{A.18}$$

Recalling the definitions of A and b, we can convert (A.18) back to matrix form (cf. [14, Equation (21)]):

$$p \in \left\{ -Y\widetilde{V}^\top L(W^\tau)^\dagger + c1_n^\top \,\middle|\, c \in \mathbb{R}^r \right\}. \tag{A.19}$$

Part (ii): Substituting the translation expression (A.19) into the Euclidean gradient with respect to Y (A.12), we obtain:

$$\begin{aligned}
\nabla_Y f(X) &= 2\left((-Y\widetilde{V}^\top L(W^\tau)^\dagger + c1_n^\top)\widetilde{V} + Y(L_{\mathrm{R}} + \widetilde{\Sigma}) \right) \\
&= 2\left(-Y\widetilde{V}^\top L(W^\tau)^\dagger \widetilde{V} + Y(L_{\mathrm{R}} + \widetilde{\Sigma}) \right) \\
&= 2YQ_{\mathrm{R}} \\
&= \nabla f_{\mathrm{R}}(Y),
\end{aligned} \tag{A.20}$$

where in the second line we again used the fact that the all-1s vector 1_n belongs to the null space of $\widetilde{V}$. Thus, we have shown that Problems 4.3 and A.1 have the same Euclidean gradient with respect to the Stiefel elements Y. Since X is a first-order critical point, the Riemannian gradient of $f(X)$ with respect to Y is zero, which implies that:

$$\operatorname{grad}_Y f(X) = \operatorname{Proj}_Y(\nabla_Y f(X)) = \operatorname{Proj}_Y(\nabla f_{\mathrm{R}}(Y)) = \operatorname{grad} f_{\mathrm{R}}(Y) = 0, \tag{A.21}$$

i.e., Y is a first-order critical point of Problem A.1. Finally, plugging the expression of p into $f(X)$ shows that $\langle Q, X^\top X\rangle = \langle Q_{\mathrm{R}}, Y^\top Y\rangle$.

Part (iii): The Lagrange multiplier matrices $\Lambda(X)$ and $\Lambda_{\mathrm{R}}(Y)$ for Problem 4.3 and Problem A.1 are:

$$\Lambda(X) = \operatorname{SymBlockDiag}_d^+\left(X^\top X Q\right) = \frac{1}{2}\operatorname{SymBlockDiag}_d^+\left(X^\top \nabla_X f(X)\right), \tag{A.22}$$

$$\Lambda_{\mathrm{R}}(Y) = \operatorname{SymBlockDiag}\left(Y^\top Y Q_{\mathrm{R}}\right) = \frac{1}{2}\operatorname{SymBlockDiag}\left(Y^\top \nabla_Y f_{\mathrm{R}}(Y)\right); \tag{A.23}$$

see (4.43a) and [14, eq. (107)], respectively. Extracting and aggregating the nonzero diagonal blocks $\bar{\Lambda}(X)$ of $\Lambda(X)$ (cf. (4.3)), we obtain:

$$\bar{\Lambda}(X) = \frac{1}{2}\operatorname{SymBlockDiag}_d\left(Y^\top \nabla_Y f(X)\right) = \frac{1}{2}\operatorname{SymBlockDiag}_d\left(Y^\top \nabla_Y f_{\mathrm{R}}(Y)\right) = \Lambda_{\mathrm{R}}(Y), \tag{A.24}$$

where we have used (A.20) for the middle equality.

Part (iv): After permutation, the certificate matrix $S(X)$ defined in (4.43b) can be written in the block form:

$$S(X) = \begin{bmatrix} L(W^\tau) & \widetilde{V} \\ \widetilde{V}^\top & L_{\mathrm{R}} + \widetilde{\Sigma} - \bar{\Lambda}(X) \end{bmatrix} \tag{A.25}$$

(cf. (A.10)). Since $L(W^\tau) \succeq 0$ and $(I - L(W^\tau)L(W^\tau)^\dagger)\widetilde{V} = 0$, it follows from [214, Thm. 4.3] that $S(X) \succeq 0$ if only if the following generalized Schur complement of

$S(X)$ with respect to $L(W^\tau)$ is positive semidefinite:

$$\Big(L_{\mathrm{R}} + \widetilde{\Sigma} - \bar{\Lambda}(X)\Big) - \widetilde{V}^\top L(W^\tau)^\dagger \widetilde{V} = Q_{\mathrm{R}} - \Lambda_{\mathrm{R}}(Y) = S_{\mathrm{R}}(Y), \tag{A.26}$$

where we have used the substitutions (A.1) and (A.7).

Part (v): We show that Problem A.1 is obtained from Problem 4.3 after analytically eliminating the translations p. Consider the problem of minimizing the cost function (A.9) with respect to translations p only (as a function of Y). Since this is an unconstrained convex quadratic problem, we can minimize this cost first with respect to p by setting the corresponding gradient to zero. In part (i) we identified the set of all translations (for a fixed Y) satisfying this condition; see equation (A.19). After replacing p in the original cost function (A.9) with any element from $\{-Y\widetilde{V}^\top L(W^\tau)^\dagger + c1_n^\top \,|\, c \in \mathbb{R}^r\}$, we obtain the following rotation-only problem,

$$\underset{Y \in \mathrm{St}(d,r)^n}{\text{minimize}} \quad \mathrm{tr}(Y(L_{\mathrm{R}} + \widetilde{\Sigma} - \widetilde{V}^\top L(W^\tau)^\dagger \widetilde{V})Y^\top) = \mathrm{tr}(Q_{\mathrm{R}} Y^\top Y), \tag{A.27}$$

which is exactly Problem A.1. This concludes our proof. □

A.1.1 Proof of Theorem 4.1

With Lemma A.1 in place, we are ready to prove the equivalence relations between our SDP relaxation (Problem 4.1) and its rotation-only version (Problem 4.2), stated in Theorem 4.1 in Chapter 4.

Proof of Theorem 4.1. We give a constructive proof where we show that from a minimizer $Z^\star \in \mathcal{S}_+^{n+dn}$ to Problem 4.1 (full SDP relaxation), we can recover a minimizer $Z_{\mathrm{R}}^\star \in \mathcal{S}_+^{dn}$ to Problem 4.2 (rotation-only SDP relaxation) with the same rank, and vice versa. Without loss of generality, let $r^\star = \mathrm{rank}(Z^\star) \geq d$. Consider the rank-$r^\star$ factorization $Z^\star = (X^\star)^\top X^\star$. Since $Z^\star$ is a feasible point for Problem 4.1, it can be readily verified that $X^\star$ is an element of the product manifold $\mathcal{M}_{\mathrm{PGO}}(r^\star, n)$ (4.4). Let $Y^\star \in \mathrm{St}(d, r^\star)^n$ be obtained by stacking all rotational components of $X^\star$. We prove that $Z_{\mathrm{R}}^\star = (Y^\star)^\top Y^\star$ is an optimal solution to Problem 4.2. To see this, first note

that $X^\star$ is an optimal solution to the rank-restricted SDP Problem 4.3. Therefore, by Lemma A.1, it holds that,

$$\langle Q, Z^\star \rangle = \langle Q, X^{\star\top} X^\star \rangle = \langle Q_{\mathrm{R}}, Y^{\star\top} Y^\star \rangle = \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^\star \rangle. \tag{A.28}$$

Now, suppose $Z_{\mathrm{R}}^\star$ is *not* an optimal solution to Problem A.1. Then there exists Z_{R}^* such that $\langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^* \rangle < \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^\star \rangle$. Once again, without loss of generality, let $\mathrm{rank}(Z_{\mathrm{R}}^*) = r^*$ and consider the rank-r factorization $Z_{\mathrm{R}}^* = Y^{*\top} Y^*$ where $Y^* \in \mathrm{St}(d, r^*)^n$. Now suppose p^* is an optimal value for translations given Y^* (see (A.19)):

$$p^* \in \left\{ -Y^* \widetilde{V}^\top L (W^\tau)^\dagger + c 1_n^\top \,\middle|\, c \in \mathbb{R}^r \right\}. \tag{A.29}$$

Let $X^* \in \mathcal{M}_{\mathrm{PGO}}(r^*, n)$ be obtained by combining Y^* and p^* and define $Z^* \triangleq X^{*\top} X^*$. Again by Lemma A.1, it holds that,

$$\langle Q, Z^* \rangle = \langle Q, X^{*\top} X^* \rangle = \langle Q_{\mathrm{R}}, Y^{*\top} Y^* \rangle = \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^* \rangle. \tag{A.30}$$

The combination of (A.28) and (A.30) would imply that $\langle Q, Z^* \rangle < \langle Q, Z^\star \rangle$, which contradicts the starting assumption that $Z^\star$ is an optimal solution. Therefore $Z_{\mathrm{R}}^\star$ must be an optimal solution with rank $r^\star$. To conclude the proof, note that using a similar argument, we can construct an optimal solution to Problem 4.1 from an optimal solution $Z_{\mathrm{R}}^\star$ to Problem 4.2 with the same rank.

□

A.1.2 Proof of Theorem 4.2

In this subsection, we formally prove the exactness guarantees of the SDP relaxation (Problem 4.1) used in this work, which is stated in Theorem 4.2 in Chapter 4.

Proof of Theorem 4.2. By [14, Proposition 2], there exists a constant as a function of the noiseless data matrix of the rotation-only SDP relaxation (Problem 4.2), denoted as $\beta \triangleq \beta(\underline{Q_{\mathrm{R}}})$, such that if $\left\| Q_{\mathrm{R}} - \underline{Q_{\mathrm{R}}} \right\|_2 < \beta$, the rotation-only SDP relaxation

(Problem 4.2) admits a unique solution $Z_{\mathrm{R}}^{\diamond} = R^{\diamond\top} R^{\diamond}$, where $R^{\diamond} \in \mathrm{SO}(d)^n$ is a globally optimal rotation estimate to PGO (Problem 2.2). Let $Z^{\star}$ be an arbitrary minimizer to Problem 4.1. By Theorem 4.1, it holds that $\langle Q, Z^{\star} \rangle = \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^{\diamond} \rangle$. Without loss of generality, let $\mathrm{rank}(Z^{\star}) = r$ where $r \geq d$. Consider the rank-r factorization $Z^{\star} = X^{\top} X$, where $X \in \mathcal{M}_{\mathrm{PGO}}(r, n)$. Note that X is a global minimizer to Problem 4.3, and hence by Lemma A.1, it holds that,

$$\langle Q, Z^{\star} \rangle = \langle Q, X^{\top} X \rangle = \langle Q_{\mathrm{R}}, Y^{\top} Y \rangle = \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^{\diamond} \rangle. \tag{A.31}$$

Above, $Y \in \mathrm{St}(d, r)^n$ extracts the Stiefel elements from X. Since Problem 4.2 admits a unique minimizer, (A.31) implies that $Y^{\top} Y$ is the same minimizer:

$$Y^{\top} Y = Z_{\mathrm{R}}^{\diamond} = R^{\diamond\top} R^{\diamond}. \tag{A.32}$$

In addition, (A.32) also implies that $\mathrm{rank}(Y) = \mathrm{rank}(Z_{\mathrm{R}}^{\diamond}) = d$. We may thus consider the d-dimensional (thin) singular value decomposition $Y = U_d \Sigma_d V_d^{\top}$. Let us define $\bar{Y} \triangleq \Sigma_d V_d^{\top}$. Since $U_d \in \mathrm{St}(d, r)$, it holds that $\bar{Y}^{\top} \bar{Y} = R^{\diamond\top} R^{\diamond}$, and therefore $\bar{Y}$ consists of n orthogonal matrices $\bar{Y} \in \mathrm{O}(d)^n$. By inspecting the first block row of this equality, we may further deduce that,

$$\bar{Y}_1^{\top} \bar{Y}_i = R_1^{\diamond\top} R_i^{\diamond}, \ \forall i \in [n]. \tag{A.33}$$

Multiplying both sides in (A.33) from the left by $U_d \bar{Y}_1$, such that the left-hand side simplifies to Y_i.

$$Y_i = U_d \bar{Y}_1 R_1^{\diamond\top} R_i^{\diamond}. \tag{A.34}$$

Let us define $A \triangleq U_d \bar{Y}_1 R_1^{\diamond\top}$. Since $U_d \in \mathrm{St}(d, r)$ and $\bar{Y}_1, R_1^{\diamond} \in \mathrm{O}(d)$, it holds that $A \in \mathrm{St}(d, r)$. Combining equality (A.34) for all i yields the following compact equation.

$$Y = A R^{\diamond}. \tag{A.35}$$

Let $p \in \mathbb{R}^{r \times n}$ contains the translations in X. Since X is a global minimizer, it

is also a first-order critical point. Therefore, we can apply part (i) of Lemma A.1 to relate p with Y:

$$p = -Y\widetilde{V}^\top L(W^\tau)^\dagger + c1_n^\top \tag{A.36}$$

$$= -AR^\diamond\widetilde{V}^\top L(W^\tau)^\dagger + c1_n^\top \qquad \text{(A.35)} \tag{A.37}$$

$$= At^\diamond + c1_n^T. \tag{A.38}$$

In the last equality (A.38), we have defined $t^\diamond \triangleq -R^\diamond\widetilde{V}^\top L(W^\tau)^\dagger$. Notice that $t^\diamond$ corresponds to a set of globally optimal translations. Finally, we note that the first block-row of the SDP solution $Z^\star = X^\top X$ may be expressed as,

$$Z^\star{}_{(1:d,:)} = (Y_1)^\top \begin{bmatrix} Y_1 & p_1 & \dots & Y_n & p_n \end{bmatrix} \tag{A.39}$$

$$= (AR_1^\diamond)^\top \begin{bmatrix} AR_1^\diamond & At_1^\diamond + c & \dots & AR_n^\diamond & At_n^\diamond + c \end{bmatrix} \tag{A.40}$$

$$= \big[\underbrace{I_d}_{R_1^\star} \quad \underbrace{R_1^{\diamond\top}t_1^\diamond + R_1^{\diamond\top}A^\top c}_{t_1^\star} \quad \dots \quad \underbrace{R_1^{\diamond\top}R_n^\diamond}_{R_n^\star} \quad \underbrace{R_1^{\diamond\top}t_n^\diamond + R_1^{\diamond\top}A^\top c}_{t_1^\star} \big]. \tag{A.41}$$

In particular, $Z^\star{}_{(1:d,:)}$ can be obtained from $(R^\diamond, t^\diamond)$ via a global rigid body transformation with rotation $R_1^{\diamond\top}$ and translation $R_1^{\diamond\top}A^\top c$. Due to the global gauge symmetry of PGO, $Z^\star{}_{(1:d,:)}$ thus also is an optimal solution.

So far, we have proved that the SDP relaxation (Problem 4.1) is exact if its rotation-only counterpart (Problem 4.2) satisfies $\left\|Q_{\mathrm{R}} - \underline{Q_{\mathrm{R}}}\right\|_2 < \beta$. To conclude the proof, let us consider the matrix M defined in (A.10) and its latent value $\underline{M}$ (*i.e.*, constructed using noiseless relative transformation measurements). Note that M and $\underline{M}$ only differ in certain blocks,

$$M = \begin{bmatrix} \underline{L}(W^\tau) & \widetilde{\underline{V}} + \Delta_{12} \\ (\widetilde{\underline{V}} + \Delta_{12})^\top & \underline{L_{\mathrm{R}}} + \widetilde{\underline{\Sigma}} + \Delta_{22} \end{bmatrix} = \underline{M} + \begin{bmatrix} 0 & \Delta_{12} \\ \Delta_{12}^\top & \Delta_{22} \end{bmatrix}. \tag{A.42}$$

Once again, the underline notation denotes the latent value of each data matrix. Matrices Δ_{12} and Δ_{22} summarize the measurement noise. Notice that the upper left block of M is *not* affected by noise, since by construction it is always the (constant)

translation-weighted graph Laplacian. Using the notation above, we can also express Q_{R} (A.1) as a function of Δ_{12} and Δ_{22},

$$Q_{\mathrm{R}}(\Delta_{12}, \Delta_{22}) = \underline{L_{\mathrm{R}}} + \widetilde{\underline{\Sigma}} + \Delta_{22} - (\widetilde{\underline{V}} + \Delta_{12})^{\top} \underline{L} (W^{\tau})^{\dagger} (\widetilde{\underline{V}} + \Delta_{12}). \tag{A.43}$$

Crucially, note that Q_{R} varies *continuously* with respect to the noise terms Δ_{12} and Δ_{22}. Therefore, as the noise tends to zero (equivalently, as M tends to $\underline{M}$), Q_{R} converges to its latent value $\underline{Q_{\mathrm{R}}}$. By definition, this guarantees the existence of a constant $\delta > 0$, such that $\|M - \underline{M}\|_2 < \delta$ implies $\left\|Q_{\mathrm{R}} - \underline{Q_{\mathrm{R}}}\right\|_2 < \beta$. Finally, since the connection Laplacian Q and M are related by a permutation of the rows and columns, it holds that,

$$\left\|Q - \underline{Q}\right\|_2 < \delta \implies \|M - \underline{M}\|_2 < \delta \implies \left\|Q_{\mathrm{R}} - \underline{Q_{\mathrm{R}}}\right\|_2 < \beta, \tag{A.44}$$

which concludes the proof. □

A.2 Convergence of RBCD and RBCD++

A.2.1 Proof of Lemma 4.1

Proof of Lemma 4.1. Our proof largely follows the proof of Lemma 3.11 in [167]. By Assumption 4.1, the reduced cost over each block f_b satisfies (4.32) globally with Lipschitz constant c_b. To simplify our notation, we drop the iteration counter k and use X_b to represent the input into BlockUpdate (Algorithm 4.4). In addition, define $g_b \triangleq \operatorname{grad} f_b(X_b)$. In the remaining proof, we also use η to represent a tangent vector in $T_{X_b}\mathcal{M}_b$. Using the simplified notation, consider the first trust-region subproblem solved in BlockUpdate (Algorithm 4.4).

$$\underset{\eta \in T_{X_b}\mathcal{M}_b}{\text{minimize}} \quad \widehat{m}_b(\eta) \triangleq f_b(X_b) + \langle \eta, g_b \rangle + \frac{1}{2}\langle \eta, H[\eta] \rangle, \tag{A.45a}$$

$$\text{subject to} \quad \|\eta\| \leq \Delta_0. \tag{A.45b}$$

Recall that Δ_0 above is the initial trust-region radius. By Assumption 4.3, there exists $c_0 \geq 0$ such that $\langle \eta, H[\eta] \rangle \leq c_0 \|\eta\|^2$ for all $\eta \in T_{X_b}\mathcal{M}_b$. Define the constant,

$$\lambda_b \triangleq \frac{1}{4} \min\left(\frac{1}{c_0}, \frac{1}{2c_b + 2c_0}\right) = \frac{1}{8(c_b + c_0)}. \tag{A.46}$$

Note that λ_b is the same constant that appears in the lower bound of the initial trust-region radius (Assumption 4.4).

We show that the required cost decrement in Lemma 4.1 is achieved by taking the so-called *Cauchy step* [167] in the first trust-region subproblem. By definition, the Cauchy step η^C minimizes the model function (A.45) along the direction of the negative Riemannian gradient, *i.e.*, $\eta^C = -\alpha^C g_b$. The step size $\alpha^C \in [0, \Delta_0 / \|g_b\|]$ can be determined in closed-form as,

$$\alpha^C = \begin{cases} \min\left(\frac{\|g_b\|^2}{\langle g_b, H[g_b]\rangle}, \frac{\Delta_0}{\|g_b\|}\right), & \text{if } \langle g_b, H[g_b]\rangle > 0, \\ \frac{\Delta_0}{\|g_b\|}, & \text{otherwise.} \end{cases} \tag{A.47}$$

A straightforward calculation (see [167, Lemma 3.7]) shows that the Cauchy step reduces the model function $\widehat{m}_b$ by at least,

$$\widehat{m}_b(0) - \widehat{m}_b(\eta^C) \geq \frac{1}{2} \min\left(\Delta_0, \frac{\|g_b\|}{c_0}\right) \|g_b\|. \tag{A.48}$$

Next, we need to relate the above model decrement (A.48) with the actual decrement of the cost function f_b. For this, we show that the following ratio,

$$\left| \frac{\widehat{f}_b(0) - \widehat{f}_b(\eta^C)}{\widehat{m}_b(0) - \widehat{m}_b(\eta^C)} - 1 \right| = \left| \frac{\widehat{m}_b(\eta^C) - \widehat{f}_b(\eta^C)}{\widehat{m}_b(0) - \widehat{m}_b(\eta^C)} \right| \tag{A.49}$$

is close to zero. Note that in (A.49) we use the fact that, by definition, $\widehat{m}_{x_b}(0) =$

$\widehat{f}_{x_b}(0) = f_b(X_b)$. We derive an upper bound on the numerator of (A.49) as follows,

$$|\widehat{m}_b(\eta^C) - \widehat{f}_b(\eta^C)| = \left|f_b(X_b) + \langle g_b, \eta^C\rangle + \frac{1}{2}\langle \eta^C, H[\eta^C]\rangle - \widehat{f}_b(\eta^C)\right| \tag{A.50}$$

$$\leq \left|f_b(X_b) + \langle g_b, \eta^C\rangle - \widehat{f}_b(\eta^C)\right| + \frac{1}{2}\left|\langle \eta^C, H[\eta^C]\rangle\right| \tag{A.51}$$

$$\leq \frac{1}{2}(c_b + c_0)\left\|\eta^c\right\|^2. \tag{A.52}$$

For the last inequality, we have used the Lipschitz-type gradient condition of f_b for the first term, and the boundedness of H for the second term. Plugging (A.48) and (A.52) into (A.49), we obtain,

$$\left|\frac{\widehat{f}_b(0) - \widehat{f}_b(\eta^C)}{\widehat{m}_b(0) - \widehat{m}_b(\eta^C)} - 1\right| \leq \frac{\frac{1}{2}(c_b + c_0)\left\|\eta^c\right\|^2}{\frac{1}{2}\min\left(\Delta_0, \frac{\|g_b\|}{c_0}\right)\|g_b\|} \qquad \text{(A.48) and (A.52)} \tag{A.53}$$

$$\leq \frac{(c_b + c_0)\Delta_0^2}{\min\left(\Delta_0, \frac{\|g_b\|}{c_0}\right)\|g_b\|} \tag{A.54}$$

$$\leq \frac{(c_b + c_0)\Delta_0}{\|g_b\|}. \tag{A.55}$$

In order to proceed, let us first impose an additional assumption that the initial trust-region radius Δ_0 is also bounded above by $\Delta_0 \leq 4\lambda_b \|g_b\|$. Towards the end of this proof, we show how this assumption can be safely removed. Under this additional assumption, it holds that $\Delta_0 \leq 4\lambda_b \|g_b\| \leq \|g_b\|/(2c_b + 2c_0)$, and thus (A.55) implies,

$$\left|\frac{\widehat{f}_b(0) - \widehat{f}_b(\eta^C)}{\widehat{m}_b(0) - \widehat{m}_b(\eta^C)} - 1\right| \leq \frac{1}{2} \implies \rho \triangleq \frac{\widehat{f}_b(0) - \widehat{f}_b(\eta^C)}{\widehat{m}_b(0) - \widehat{m}_b(\eta^C)} \geq \frac{1}{2}. \tag{A.56}$$

In particular, ρ is bigger than the acceptance threshold of $1/4$ in BlockUpdate (see Algorithm 4.4), and thus η^C is *guaranteed* to be accepted. Consider the candidate solution corresponding to the Cauchy step $X_b' = \mathrm{Retr}_{X_b}(\eta^C)$. Using the definition of

the pullback function, it holds that,

$$f_b(X_b) - f_b(X_b') = \widehat{f}_b(0) - \widehat{f}_b(\eta^C) \tag{A.57}$$

$$\geq \frac{1}{2}\left(\widehat{m}_b(0) - \widehat{m}_b(\eta^C)\right) \tag{A.58}$$

$$\geq \frac{1}{4}\min\left(\Delta_0, \frac{\|g_b\|}{c_0}\right)\|g_b\| \tag{A.59}$$

$$\geq \frac{1}{4}\min\left(\lambda_b, \frac{1}{c_0}\right)\|g_b\|^2 \tag{A.60}$$

$$= \frac{1}{4}\lambda_b \|g_b\|^2 . \tag{A.61}$$

Therefore, we have proven that under the additional assumption that $\Delta_0 \leq 4\lambda_b \|g_b\|$, the desired cost reduction can be achieved simply by taking the Cauchy step in the first trust-region subproblem. As we mention in Section 4.3, in practice, we use the truncated conjugate-gradient (tCG) algorithm to improve upon the initial Cauchy step. However, since each additional tCG iteration will strictly decrease the model function (see [17, Proposition 7.3.2]), we can show that the inequality (A.55) holds at all times, and thus the desired cost reduction is always achieved.

To complete the proof, we need to show that Algorithm 4.4 still achieves the desired cost reduction, even after removing the additional assumption that $\Delta_0 \leq 4\lambda_b \|g_b\|$. To do so, note that after dropping this assumption, inequality (A.56) might fail to hold and as a result the Cauchy step can be rejected in the first iteration. However, by the mechanism of BlockUpdate (Algorithm 4.4), after each rejection the trust-region radius will be divided by four in the next iteration. Therefore, in the worst case, the trust-region radius will be within the interval $[\lambda_b \|g_b\|, 4\lambda_b \|g_b\|]$ after $\mathcal{O}(\log(4\lambda_b \|g_b\| \Delta_0))$ consecutive rejections, after which the Cauchy step is guaranteed to be accepted in the next trust-region subproblem.

□

A.2.2 Proof of Theorem 4.4

Proof of Theorem 4.4. We begin by using the sufficient descent property proved in Lemma 4.1, which states that after each BLOCKUPDATE operation, the reduced cost is decreased by at least,

$$f_{b_k}(X_{b_k}^k) - f_{b_k}(X_{b_k}^{k+1}) \geq \frac{1}{4}\lambda_{b_k} \left\|\text{grad}\, f_{b_k}(X_{b_k}^k)\right\|^2 = \frac{1}{4}\lambda_{b_k} \left\|\text{grad}_{b_k} f(X^k)\right\|^2. \quad \text{(A.62)}$$

Recall that by design of the RBCD algorithm, at each iteration the values of all blocks other than b_k remain unchanged. Denote the values of these fixed blocks as X_c^k. By definition of the reduced cost f_{b_k}, it is straightforward to verify that the decrease in f_{b_k} exactly equals the decrease in the global cost f,

$$f_{b_k}(X_{b_k}^k) - f_{b_k}(X_{b_k}^{k+1}) = f(X_{b_k}^k, X_c^k) - f(X_{b_k}^{k+1}, X_c^k) = f(X^k) - f(X^{k+1}). \quad \text{(A.63)}$$

The above result directly implies that each iteration of RBCD reduces the global cost function by at least,

$$f(X^k) - f(X^{k+1}) = f_{b_k}(X_{b_k}^k) - f_{b_k}(X_{b_k}^{k+1}) \geq \frac{1}{4}\lambda_{b_k} \left\|\text{grad}_{b_k} f(X^k)\right\|^2. \quad \text{(A.64)}$$

In the remaining proof, we use (A.64) to prove the stated convergence rate for each block selection rule.

Uniform Sampling. In this case, the updated block $b_k \in [N]$ is selected uniformly at random at each iteration. Conditioned on all previously selected blocks $b_{0:k-1}$, we can take expectation on both sides of (A.64) with respect to the current block b_k.

$$\mathbb{E}_{b_k|b_{0:k-1}}\Big[f(X^k) - f(X^{k+1})\Big] \geq \sum_{b\in N} \text{Prob}(b_k = b) \cdot \frac{1}{4}\lambda_b \left\|\text{grad}_b\, f(X^k)\right\|^2 \quad \text{(A.65)}$$

Recall that all block are selected with equal probability, *i.e.*, $\text{Prob}(b_k = b) = 1/N$ for all $b \in [N]$. Using this, we arrive at a simplified inequality that lower bounds the

expected cost decrease by the squared gradient norm with respect to all variables X.

$$\mathbb{E}_{b_k|b_{0:k-1}}\Big[f(X^k)-f(X^{k+1})\Big] \geq \sum_{b\in N}\frac{1}{N}\cdot\frac{1}{4}\lambda_b\left\|\text{grad}_b\, f(X^k)\right\|^2 \geq \frac{\min_{b\in[N]}\lambda_b}{4N}\left\|\text{grad}\, f(X^k)\right\|^2. \tag{A.66}$$

We can use the above inequality (A.66) to bound the expected cost decrease over all iterations from $k=0$ to $k=K$. To do so, we first write the overall cost reduction as a cascading sum, and then apply the law of total expectation on each expectation term that appears inside the sum.

$$\begin{aligned} f(X^0)-\mathbb{E}_{b_{0:K-1}} f(X^K) &= \sum_{k=0}^{K-1}\mathbb{E}_{b_{0:k}}\Big[f(X^k)-f(X^{k+1})\Big] \\ &= \sum_{t=0}^{K-1}\mathbb{E}_{b_{0:k-1}}\Big[\mathbb{E}_{b_k|b_{0:k-1}}[f(X^k)-f(X^{k+1})]\Big]. \end{aligned} \tag{A.67}$$

Observe that each innermost conditional expectation is already bounded by our previous inequality (A.66). Plugging in this lower bound, we arrive at,

$$f(X^0)-\mathbb{E}_{b_{0:K-1}} f(X^K) \geq \sum_{k=0}^{K-1}\mathbb{E}_{b_{0:k-1}}\left[\frac{\min_{b\in[N]}\lambda_b}{4N}\left\|\text{grad}\, f(X^k)\right\|^2\right] \tag{A.68}$$

$$\geq K\cdot\frac{\min_{b\in[N]}\lambda_b}{4N}\min_{0\leq k\leq K-1}\mathbb{E}_{b_{0:k-1}}\left\|\text{grad}\, f(X^k)\right\|^2. \tag{A.69}$$

To conclude the proof, we note that since $f^\star$ is the global minimum, we necessarily have $f(X^0)-f^\star \geq f(X^0)-\mathbb{E}_{b_{0:K-1}} f(X^K)$. This directly implies,

$$f(X^0)-f^\star \geq K\cdot\frac{\min_{b\in[N]}\lambda_b}{4N}\min_{0\leq k\leq K-1}\mathbb{E}_{b_{0:k-1}}\left\|\text{grad}\, f(X^k)\right\|^2. \tag{A.70}$$

Rearranging the last inequality gives the desired convergence rate (4.38).

Importance Sampling. We start with the same procedure of taking conditional expectations on both sides of (A.64). The only difference is that with importance sampling, at each iteration the block is selected with probability proportional to the squared gradient norm, *i.e.*, $p_b = \left\|\text{grad}_b\, f(X^k)\right\|^2 / \left\|\text{grad}\, f(X^k)\right\|^2$. Using this to

expand the conditional expectation gives:

$$\mathbb{E}_{b_k|b_{0:k-1}}[f(X^k) - f(X^{k+1})] \geq \sum_{b\in[N]} \frac{\left\|\text{grad}_b\, f(X^k)\right\|^2}{\left\|\text{grad}\, f(X^k)\right\|^2} \cdot \frac{1}{4}\lambda_b \left\|\text{grad}_b\, f(X^k)\right\|^2 \tag{A.71}$$

$$\geq \frac{\min_{b\in[N]} \lambda_b}{4} \cdot \frac{\sum_{b\in[N]} \left\|\text{grad}_b\, f(X^k)\right\|^4}{\sum_{b\in[N]} \left\|\text{grad}_b\, f(X^k)\right\|^2} \tag{A.72}$$

$$= \frac{\min_{b\in[N]} \lambda_b}{4} \cdot \frac{\sum_{b\in[N]} a_b^2}{\sum_{b\in[N]} a_b}, \tag{A.73}$$

where we define $a_b \triangleq \left\|\text{grad}_b\, f(X^k)\right\|^2$ for brevity. Applying the Cauchy-Schwarz inequality gives,

$$\frac{1}{N^2}\Big(\sum_{b\in[N]} a_b\Big)^2 \leq \frac{1}{N}\sum_{b\in[N]} a_b^2. \tag{A.74}$$

Rearranging the above inequality yields,

$$\frac{\sum_{b\in[N]} a_b^2}{\sum_{b\in[N]} a_b} \geq \frac{1}{N}\sum_{b\in[N]} a_b = \frac{1}{N}\left\|\text{grad}\, f(X^k)\right\|^2. \tag{A.75}$$

Combining (A.73) and (A.75) gives,

$$f(X^k) - \mathbb{E}_{b_k|b_{0:k-1}} f(X^{k+1}) \geq \frac{\min_{b\in[N]} \lambda_b}{4N}\left\|\text{grad}\, f(X^k)\right\|^2. \tag{A.76}$$

The rest of the proof is identical to the proof for uniform sampling.

Greedy Selection. With greedy selection, we can perform a deterministic analysis. Recall that at each iteration, the block with the largest squared gradient norm is selected. Using this information inside our inequality (A.64), we arrive at,

$$f(X^k) - f(X^{k+1}) \geq \frac{1}{4}\lambda_{b_k}\left\|\text{grad}_{b_k}\, f(X^k)\right\|^2 \tag{A.77}$$

$$\geq \frac{1}{4}\left(\min_{b\in[N]} \lambda_b\right) \cdot \left(\max_{b\in[N]} \left\|\text{grad}_b\, f(X^k)\right\|^2\right) \tag{A.78}$$

$$\geq \frac{1}{4N}\min_{b\in[N]} \lambda_b \cdot \left\|\text{grad}\, f(X^k)\right\|^2. \tag{A.79}$$

A telescoping sum of the above inequalities from $k = 0$ to $k = K - 1$ gives,

$$f(X^0) - f^\star \geq \sum_{k=0}^{K-1} [f(X^k) - f(X^{k+1})] \tag{A.80}$$

$$\geq \sum_{k=0}^{K-1} \frac{1}{4N} \min_{b\in[N]} \lambda_b \cdot \left\|\operatorname{grad} f(X^k)\right\|^2 \tag{A.81}$$

$$\geq \frac{K}{4N} \cdot \min_{b\in[N]} \lambda_b \cdot \min_{0\leq k\leq K-1} \left\|\operatorname{grad} f(X^k)\right\|^2. \tag{A.82}$$

Rearranging the last inequality gives (4.39). □

A.2.3 Proof of Theorem 4.5

Proof. For the purpose of proving global first-order convergence, we only focus on the evolution of the $\{X^k\}$ sequence in RBCD++ (Algorithm 4.5). After each iteration, there are two possibilities depending on whether the adaptive restart condition (line 13) is triggered. If restart is not triggered, then by construction the current iteration must achieve a cost reduction that is at least,

$$f(X^k) - f(X^{k+1}) \geq c_1 \left\|\operatorname{grad}_{b_k} f(X^k)\right\|^2. \tag{A.83}$$

On the other hand, if restart is triggered, then the algorithm would instead update the selected block using the default BlockUpdate method. In this case, using the same argument as the beginning of proof for Theorem 4.4, we can establish the following lower bound on global cost reduction.

$$f(X^k) - f(X^{k+1}) = f_{b_k}(X^k_{b_k}) - f_{b_k}(X^{k+1}_{b_k}) \geq \frac{1}{4}\lambda_{b_k} \left\|\operatorname{grad}_{b_k} f(X^k)\right\|^2. \tag{A.84}$$

Combining the two cases, we see that each iteration of RBCD++ decreases the global cost by at least,

$$f(X^k) - f(X^{k+1}) \geq \min\left(c_1, \lambda_{b_k}/4\right) \left\|\mathrm{grad}_{b_k} f(X^k)\right\|^2 \tag{A.85}$$

$$\geq \min\left(c_1, \min_{b\in[N]} \lambda_b/4\right) \left\|\mathrm{grad}_{b_k} f(X^k)\right\|^2 \tag{A.86}$$

$$\geq C \left\|\mathrm{grad}_{b_k} f(X^k)\right\|^2. \tag{A.87}$$

The above inequality serves as a lower bound on the global cost reduction after each iteration. Note that this lower bound is very similar to (A.64) used in the proof of Theorem 4.4. The only difference is the change of constant on the right-hand side. From this point on, we can employ the same steps used in the proof of Theorem 4.4 to prove the desired convergence rates for all three block selection rules. □

A.2.4 Convergence on Problem 4.3

In this section, we address the convergence of RBCD and RBCD++ when solving the rank-restricted semidefinite relaxations of PGO (Problem 4.3). To apply the general convergence results in Section 4.4, we need to check that the required technical conditions are satisfied. Among these, we note that the Riemannian Hessian operator (which we use as the user-specified map H in Algorithm 4.4) satisfies Assumption 4.2 due to the linearity of affine connections [17, Section 5.2]. In addition, Assumption 4.4 can be satisfied by choosing a sufficiently large initial trust-region radius. Therefore, to establish convergence, we are left to verify Assumption 4.1 (Lipschitz-type gradients for pullback) and Assumption 4.3 (boundedness of Riemannian Hessian).

Verifying Assumption 4.1

We now show that Assumption 4.1 is satisfied in Problem 4.3, *i.e.*, the reduced cost function (4.21) corresponding to any block has Lipschitz-type gradient for pullbacks along the iterates of RBCD and RBCD++. In [167], Boumal *et al.* prove a simple condition for a function $f : \mathcal{M} \to \mathbb{R}$ defined on a matrix submanifold to have

Lipschitz-type gradient for pullbacks. For convenience, we include their result below.

Lemma A.2 (Lemma 2.7 in [167]). *Let $\mathcal{E}$ be a Euclidean space and let $\mathcal{M}$ be a compact Riemannian submanifold of $\mathcal{E}$. Let* Retr *be a retraction on $\mathcal{M}$ (globally defined). If $f : \mathcal{E} \to \mathbb{R}$ has Lipschitz continuous gradient then the pullbacks $f \circ \mathrm{Retr}_x$ satisfy* (4.32) *globally with some constant c_g independent of x.*

In our case, we need a generalized version of Lemma A.2 that extend the result to product manifold with Euclidean spaces.

Lemma A.3 (Extension of Lemma A.2 to product manifolds with Euclidean spaces). *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be Euclidean spaces, and define $\mathcal{E} \triangleq \mathcal{E}_1 \times \mathcal{E}_2$. Let $\mathcal{M} \triangleq \mathcal{M}_1 \times \mathcal{E}_2$, where $\mathcal{M}_1$ is a compact Riemannian submanifold of $\mathcal{E}_1$. Given $x = [x_1\ x_2] \in \mathcal{M}$ and $\eta = [\eta_1\ \eta_2] \in T_x\mathcal{M}$, define a retraction operator $\mathrm{Retr}_x : T_x\mathcal{M} \to \mathcal{M}$ as: $\mathrm{Retr}_x(\eta) = [\mathrm{Retr}_{x_1}(\eta_1)\ \mathrm{Retr}_{x_2}(\eta_2)] = [\mathrm{Retr}_{x_1}(\eta_1)\ x_2 + \eta_2]$, where Retr_{x_1} is a globally defined retraction on $\mathcal{M}_1$ and we employ the standard retraction for Euclidean space. If $f : \mathcal{E} \to \mathbb{R}$ has bounded and Lipschitz continuous gradient with Euclidean Lipschitz constant c_l, then the pullbacks $f \circ \mathrm{Retr}_x$ satisfy* (4.32) *globally with some constant $c_g \geq c_l$ independent of x;* i.e.,

$$\left|\widehat{f}_x(\eta) - [f(x) + \langle \eta, \mathrm{grad}_x f \rangle]\right| \leq \frac{c_g}{2} \|\eta\|^2 . \tag{A.88}$$

Proof. This proof is a straightforward generalization of the proof of Lemma A.2. By assumption, the Euclidean gradient ∇f is Lipschitz continuous with Lipschitz constant c_l , which implies that for any $x, y \in \mathcal{M}$,

$$|f(y) - [f(x) + \langle \nabla_x f, y - x \rangle]| \leq \frac{c_l}{2} \|y - x\|^2 . \tag{A.89}$$

The above inequality is true in particular for any $y = \mathrm{Retr}_x(\eta), \eta \in T_x\mathcal{M}$. In this

case, the inner product that appears in the LHS of (A.89) can be expanded as,

$$\langle \nabla_x f, \mathrm{Retr}_x(\eta) - x \rangle = \left\langle \begin{bmatrix} \nabla_{x_1} f & \nabla_{x_2} f \end{bmatrix}, \begin{bmatrix} \mathrm{Retr}_{x_1}(\eta_1) - x_1 & (x_2 + \eta_2) - x_2 \end{bmatrix} \right\rangle \quad \text{(A.90)}$$

$$= \langle \nabla_{x_1} f, \mathrm{Retr}_{x_1}(\eta_1) - x_1 \rangle + \langle \nabla_{x_2} f, \eta_2 \rangle \quad \text{(A.91)}$$

$$= \langle \nabla_{x_1} f, \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 + \eta_1 \rangle + \langle \nabla_{x_2} f, \eta_2 \rangle \quad \text{(A.92)}$$

$$= \langle \nabla_{x_1} f, \eta_1 \rangle + \langle \nabla_{x_2} f, \eta_2 \rangle + \langle \nabla_{x_1} f, \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \rangle. \quad \text{(A.93)}$$

Next, we use two facts: (1) the Riemannian gradient in Euclidean space is just the standard (Euclidean) gradient; and (2) the Riemannian gradient of submanifolds *embedded* in a Euclidean space is the orthogonal projection of the Euclidean gradient onto the tangent space ([17, Equation 3.37]). With these, the above equality can be further simplified to,

$$\langle \nabla_x f, \mathrm{Retr}_x(\eta) - x \rangle = \langle \mathrm{grad}_{x_1} f, \eta_1 \rangle + \langle \mathrm{grad}_{x_2} f, \eta_2 \rangle + \langle \nabla_{x_1} f, \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \rangle \quad \text{(A.94)}$$

$$= \langle \mathrm{grad}_x f, \eta \rangle + \langle \nabla_{x_1} f, \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \rangle. \quad \text{(A.95)}$$

Plugging (A.95) into (A.89) gives,

$$\begin{aligned} & \left| f(y) - [f(x) + \langle \nabla_x f, y - x \rangle] \right| \\ = & \left| f(\mathrm{Retr}_x(\eta)) - [f(x) + \langle \mathrm{grad}_x f, \eta \rangle + \langle \nabla_{x_1} f, \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \rangle] \right| \\ \leq & \frac{c_l}{2} \left\| \mathrm{Retr}_x(\eta) - x \right\|^2. \end{aligned} \quad \text{(A.96)}$$

Applying the triangle and Cauchy-Schwarz inequalities and expanding $\|\mathrm{Retr}_x(\eta) - x\|^2$

yields,

$$
\begin{aligned}
&|f(\mathrm{Retr}_x(\eta)) - [f(x) + \langle \mathrm{grad}_x f, \eta \rangle]| \\
\leq &\frac{c_l}{2} \left\| \mathrm{Retr}_x(\eta) - x \right\|^2 + |\langle \nabla_{x_1} f, \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \rangle| && \text{(A.97)} \\
\leq &\frac{c_l}{2} \left\| \mathrm{Retr}_x(\eta) - x \right\|^2 + \left\| \nabla_{x_1} f \right\| \left\| \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \right\| && \text{(A.98)} \\
= &\frac{c_l}{2} \left\| \eta_2 \right\|^2 + \frac{c_l}{2} \left\| \mathrm{Retr}_{x_1}(\eta_1) - x_1 \right\|^2 + \left\| \nabla_{x_1} f \right\| \left\| \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \right\|. && \text{(A.99)}
\end{aligned}
$$

Since we assume that the Euclidean gradient is bounded, there exists constant $G_1 \geq 0$ such that $\left\| \nabla_{x_1} f \right\| \leq G_1$ for all $x_1 \in \mathcal{M}_1$. In equations (B.3) and (B.4) in [167], Boumal *et al.* show that for the compact submanifold $\mathcal{M}_1$, the following inequalities hold,

$$
\left\| \mathrm{Retr}_{x_1}(\eta_1) - x_1 \right\| \leq \alpha_1 \left\| \eta_1 \right\|, \tag{A.100}
$$

$$
\left\| \mathrm{Retr}_{x_1}(\eta_1) - x_1 - \eta_1 \right\| \leq \beta_1 \left\| \eta_1 \right\|^2, \tag{A.101}
$$

for some $\alpha_1, \beta_1 \geq 0$. Plugging (A.100) and (A.101) in (A.99) gives,

$$
|f(\mathrm{Retr}_x(\eta)) - [f(x) + \langle \mathrm{grad}_x f, \eta \rangle]| \leq \frac{c_l}{2} \left\| \eta_2 \right\|^2 + \frac{c_l}{2} \alpha_1^2 \left\| \eta_1 \right\|^2 + G_1 \beta_1 \left\| \eta_1 \right\|^2. \tag{A.102}
$$

Let us define c_g as follows,

$$
c_g \triangleq \max \Big(c_l, c_l \alpha_1^2 + 2 G_1 \beta_1 \Big) \geq c_l. \tag{A.103}
$$

Finally, combining (A.102) and (A.103) yields the desired result,

$$
|f(\mathrm{Retr}_x(\eta)) - [f(x) + \langle \mathrm{grad}_x f, \eta \rangle]| \leq c_g (\left\| \eta_1 \right\|^2 + \left\| \eta_2 \right\|^2) = \frac{c_g}{2} \left\| \eta \right\|^2. \tag{A.104}
$$

□

Let us consider the reduced cost f_b (4.21) as a function defined on the product manifold $\mathrm{St}(d, r)^{n_b} \times \mathbb{R}^{r \times n_b}$, where n_b is the number of poses contained in block b.

Note that in the ambient (Euclidean) space, f_b is a quadratic function of X_b. Hence, its Euclidean gradient $\nabla f_b(X_b)$ is Lipschitz continuous, with Lipschitz constant given by the maximum eigenvalue of Q_b. Next, we show that within any sublevel set of the global cost function f, $\nabla f_b(X_b)$ is also bounded for any block $b \in [N]$. Since $\nabla f_b(X_b) = \nabla_b f(X)$ by construction of the reduced cost function, it is equivalent to bound the global Euclidean gradient $\nabla f(X)$. Write out the individual rotation and translation components of $X \in \mathcal{M}_{\text{PGO}}(r, n)$,

$$X = \begin{bmatrix} Y_1 & p_1 & \dots & Y_n & p_n \end{bmatrix}. \tag{A.105}$$

For any X in the $\bar{f}$-sublevel-set, expanding $f(X) \leq \bar{f}$ yields,

$$f(X) = \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \left\| Y_j - Y_i \widetilde{R}_{ij} \right\|_F^2 + \sum_{(i,j)\in\mathcal{E}} \tau_{ij} \left\| p_j - p_i - Y_i \widetilde{t}_{ij} \right\|_2^2 \leq \bar{f}. \tag{A.106}$$

Since the objective consists of a sum over squared terms, we can obtain a *uniform* upper bound on each single term. Specifically, for each edge $(i, j) \in \mathcal{E}$, the associated relative translation cost is bounded by,

$$\left\| p_j - p_i - Y_i \widetilde{t}_{ij} \right\|_2^2 \leq \bar{f} / \min_{(i,j)\in\mathcal{E}} \tau_{ij} \implies \left\| p_j - p_i - Y_i \widetilde{t}_{ij} \right\|_2 \leq \sqrt{\bar{f} / \min_{(i,j)\in\mathcal{E}} \tau_{ij}}. \tag{A.107}$$

Note that when forming $\nabla f(X)$, the only terms that are potentially unbounded correspond to the translation terms $h_{ij} \triangleq \tau_{ij} \left\| p_j - p_i - Y_i \widetilde{t}_{ij} \right\|_2^2$, since the rotation terms are defined on compact search spaces. The Euclidean gradients of h_{ij} with respect to Y_i, p_i, p_j are given by,

$$\nabla_{Y_i} h_{ij} = -2(p_j - p_i - Y_i \widetilde{t}_{ij}) \widetilde{t}_{ij}^\top, \tag{A.108}$$

$$\nabla_{p_i} h_{ij} = -2(p_j - p_i - Y_i \widetilde{t}_{ij}), \tag{A.109}$$

$$\nabla_{p_j} h_{ij} = 2(p_j - p_i - Y_i \widetilde{t}_{ij}), \tag{A.110}$$

By (A.107), the norms of (A.108)-(A.110) are bounded, which implies that the overall Euclidean gradient $\nabla f(X)$ is bounded within the sublevel set. Applying Lemma A.3,

we have shown that f_b has Lipschitz-type gradient for pullbacks within the sublevel set of f.

Finally, note that both RBCD and RBCD++ are by construction *descent methods*, *i.e.*, the sequence of iterates satisfies $f(X^k) \leq f(X^{k-1}) \leq \ldots \leq f(X^1) \leq f^0$ for any k. For RBCD, this is true since we explicitly enforce that each block update should produce a function decrease that is at least a constant fraction of the model decrease (Algorithm 4.4, line 5). For RBCD++, this is also guaranteed by the adaptive restart condition (see Section 4.3.3). This property ensures that the sequence of iterates generated by RBCD or RBCD++ never leaves the initial sublevel set, and therefore Assumption 4.1 is satisfied.

Verifying Assumption 4.3

We can follow a similar strategy and show that the Riemannian Hessian operator is bounded at any iterate generated by RBCD and RBCD++, hence satisfying Assumption 4.3. Let f denote the cost function in Problem 4.3, and $X^k \in \mathcal{M}_{\text{PGO}}(r, n)$ denote the solution at iteration k. From (2.3), it may be straightforwardly verified that for Problem 4.3, the Riemannian Hessian operator has the following explicit form:

$$\operatorname{Hess} f(X)[\eta] = 2 \operatorname{Proj}_{T_X}(\eta S(X)). \tag{A.111}$$

Above, $S(X)$ is the "dual certificate" matrix (4.43b) defined in Theorem 4.6, repeated below for convenience,

$$S(X) \triangleq Q - \operatorname{SymBlockDiag}_d^+(X^\top X Q). \tag{A.112}$$

In particular, note that (A.111) implies the following equality,

$$\langle \eta, \operatorname{Hess} f(X)[\eta] \rangle = 2\langle \eta, S(X)\eta \rangle. \tag{A.113}$$

Following the strategy in Section A.2.4, we now show that $S(X)$ is bounded within the sublevel sets of f. To make the presentation clear, we present the proof in three

steps.

Step 1: Eliminating Global Translation Symmetry from $S(X)$. To begin, we observe that the connection Laplacian Q in Problem 4.3 always has a null vector [20],

$$v_0 \triangleq 1_n \otimes \begin{bmatrix} 0_d \\ 1 \end{bmatrix}, \tag{A.114}$$

where 1_n stands for the vector of all ones. Intuitively, v_0 arises as a result of the global translation symmetry inherent in PGO. We note that although v_0 is a single vector, it actually accounts for the *complete* set of translational symmetries which has dimension r. More precisely, adding a multiple of v_0 on the k-th row of X ($1 \leq k \leq r$) corresponds to applying a global translation along the k-th unit direction. Let us consider projecting each row of our decision variable X onto the subspace orthogonal to v_0. In matrix notation, this operation can be written as a matrix product XP, where P is a projection matrix defined as follows.

$$P \triangleq I - v_0 {v_0}^\top / \|v_0\|_2^2. \tag{A.115}$$

Note that after projection, it holds that $X = XP + uv_0^\top$ for some vector $u \in \mathbb{R}^r$. Substitute this decomposition into the second term of (A.112),

$$\begin{aligned}
\mathrm{SymBlockDiag}_d^+(X^\top XQ) &= \mathrm{SymBlockDiag}_d^+((XP + uv_0^\top)^\top (XP + uv_0^\top)Q) && \text{(A.116)} \\
&= \mathrm{SymBlockDiag}_d^+((XP + uv_0^\top)^\top XPQ) && \text{(A.117)} \\
&= \mathrm{SymBlockDiag}_d^+((XP)^\top XPQ) \\
&\quad + \mathrm{SymBlockDiag}_d^+(\underbrace{v_0 u^\top XPQ}_{A}). && \text{(A.118)}
\end{aligned}$$

In (A.118), we have used the linearity of the $\mathrm{SymBlockDiag}_d^+$ operator (4.3). Consider A defined in (A.118) as a $(d+1)$-by-$(d+1)$ block-structured matrix. Using the special structure of v_0, we can verify that for each diagonal block of A, its top-left d-by-d submatrix is always zero. Thus, applying $\mathrm{SymBlockDiag}_d^+$ zeros out this term and it

holds that,

$$S(X) = Q\text{−SymBlockDiag}_d^+(X^\top XQ) = Q\text{−SymBlockDiag}_d^+((XP)^\top XPQ) = S(XP). \tag{A.119}$$

Equation (A.119) proves the intuitive result that the dual certificate matrix $S(X)$ is invariant to any global translations applied to X.

Step 2: Bounding $S(X)$ within the sublevel set of f. We now prove that for all X for which $f(X) \leq \bar{f}$, the spectral norm $\|S(X)\|_2$ is upper bounded by a constant which only depends on $\bar{f}$. In light of (A.119), for the purpose of bounding $S(X)$, we can assume without loss of generality that $X = XP$. This simply means that each row of X is orthogonal to v_0, which implies that $\sum_{i=1}^n p_i = 0$, *i.e.*, the translations are *centered* at zero. Recall the bound (A.107) we have obtained for translation terms in the previous section. Using triangle inequality, we can move $Y_i\widetilde{t}_{ij}$ to the right-hand side, so that we obtain an upper bound on the relative translation between p_i and p_j,

$$\|p_j - p_i\|_2 \leq \sqrt{\bar{f}/\min_{(i,j)\in\mathcal{E}} \tau_{ij}} + \left\|Y_i\widetilde{t}_{ij}\right\|_2 \tag{A.120}$$

$$= \sqrt{\bar{f}/\min_{(i,j)\in\mathcal{E}} \tau_{ij}} + \left\|\widetilde{t}_{ij}\right\|_2 \tag{A.121}$$

$$\leq \sqrt{\bar{f}/\min_{(i,j)\in\mathcal{E}} \tau_{ij}} + \max_{(i,j)\in\mathcal{E}} \left\|\widetilde{t}_{ij}\right\|_2. \tag{A.122}$$

Taking the square of the above bound and summing over all edges in the pose graph, it holds that,

$$\sum_{(i,j)\in\mathcal{E}} \|p_j - p_i\|_2^2 \leq |\mathcal{E}|\left(\sqrt{\bar{f}/\min_{(i,j)\in\mathcal{E}} \tau_{ij}} + \max_{(i,j)\in\mathcal{E}} \left\|\widetilde{t}_{ij}\right\|_2\right)^2 \triangleq c_f. \tag{A.123}$$

Note that the constant c_f only depends on $\bar{f}$ and the input measurements. Let $\text{vec}(p) \in \mathbb{R}^{rn}$ be the vector formed by concatenating all $p_1, \ldots, p_n$. The left-hand side of (A.123) can be written in matrix form as $\text{vec}(p)^\top(L \otimes I_r)\text{vec}(p)$, where L is the (unweighted) Laplacian of the pose graph $G = (\mathcal{V}, \mathcal{E})$. Since G is connected, the Laplacian has a rank-1 null space spanned by the all-ones vector 1_n. Correspondingly,

$L \otimes I_r$ has a rank-r null space spanned by the columns of $1_n \otimes I_r$. Crucially, using our assumption that $\sum_{i=1}^n p_i = 0$, it can be readily checked that $\mathrm{vec}(p)$ is *orthogonal* to this null space. Therefore, we can obtain a lower bound on $\mathrm{vec}(p)^\top (L \otimes I_r)\mathrm{vec}(p)$ using the smallest *positive* eigenvalue of $L \otimes I_r$, which coincides with the *algebraic connectivity* of G, denoted as $\lambda_2(L)$,

$$\mathrm{vec}(p)^\top (L \otimes I_r)\mathrm{vec}(p) \geq \lambda_2(L) \left\|\mathrm{vec}(p)\right\|_2^2. \tag{A.124}$$

Combine this inequality with (A.123), we have thus shown that,

$$\lambda_2(L) \left\|\mathrm{vec}(p)\right\|_2^2 \leq c_f. \tag{A.125}$$

Since $\lambda_2(L)$ is guaranteed to be positive as the graph is connected, we can divide both sides by $\lambda_2(L)$. After taking the square root, we obtain the following bound on the translations,

$$\left\|\mathrm{vec}(p)\right\|_2 \leq \sqrt{c_f/\lambda_2(L)}. \tag{A.126}$$

Recall that X contains translations in addition to n "lifted" rotations $Y_1, \ldots, Y_n \in \mathrm{St}(d, r)$. Since Y_i is an element of the Stiefel manifold, the norm of Y_i in the ambient space is always a constant. This implies that the Frobenius norm $\left\|X\right\|_F$ is bounded. Finally, using the fact that $S(X)$ as defined in (A.112) is a continuous operator together with (A.113), it holds that the induced operator norm of $\mathrm{Hess}\, f(X)$ on the tangent space is bounded within the sublevel set.

Step 3: Bounding the Riemannian Hessian along iterates of RBCD and RBCD++. Once again, since both RBCD and RBCD++ are descent methods by construction, any sequence of iterates will remain in the initial sublevel set. Therefore, by the result obtained in **Step 2**, there exists a constant c_0 (whose value only depends on $f(X^0)$) that bounds the Riemannian Hessian at all iterations:

$$\max_{\eta \in T_{X^k}\mathcal{M}_{\mathrm{PGO}}(r,n), \|\eta\|=1} \left|\langle \eta, \mathrm{Hess}\, f(X^k)[\eta]\rangle\right| \leq c_0, \ \forall k. \tag{A.127}$$

To complete the proof, we show that the reduced Riemannian Hessian, Hess $f_b(X_b^k)$, at an arbitrary block b is also bounded by c_0. This is needed as during distributed local search, we apply BLOCKUPDATE (Algorithm 4.4) on individual blocks instead of the full problem. Suppose for the sake of contradiction that there exists a tangent vector $\eta_b \in T_{X_b^k}\mathcal{M}_{\text{PGO}}(r, n_b)$ such that $\|\eta_b\| = 1$ and $|\langle \eta_b, \text{Hess}\, f_b(X_b^k)[\eta_b]\rangle| > c_0$. Let $\gamma_b : [-\epsilon, \epsilon] \to \mathcal{M}_{\text{PGO}}(r, n_b)$ be the corresponding geodesic such that $\gamma_b(0) = X_b$ and $\gamma_b'(0) = \eta_b$. Define the scalar function $h_b \triangleq f_b \circ \gamma_b$. By standard results in differential geometry (*e.g.*, see [17]), it holds that $h_b''(0) = \langle \eta_b, \text{Hess}\, f_b(X_b^k)[\eta_b]\rangle$.

Using the product structure of our manifold, we can associate γ_b with a corresponding geodesic $\gamma : [-\epsilon, \epsilon] \to \mathcal{M}_{\text{PGO}}(r, n)$ on the overall manifold such that γ agrees with γ_b at the selected block coordinate b and stays constant at all other blocks. Consider the tangent vector $\eta \in T_{\gamma(0)}\mathcal{M}_{\text{PGO}}(r, n)$. By construction, the blocks of η are given by,

$$\eta_{b'} = \begin{cases} \eta_b, & \text{if } b' = b, \\ 0, & \text{otherwise.} \end{cases} \tag{A.128}$$

As a result, we have $\|\eta\| = 1$. Define the scalar function $h \triangleq f \circ \gamma$ and we have by construction that $h''(0) = h_b''(0)$. This would then imply,

$$|h_b''(0)| = |h''(0)| = |\langle \eta, \text{Hess}\, f(X^k)[\eta]\rangle| > c_0, \tag{A.129}$$

which is a contradiction.

A.3 Proof of Theorem 4.3

In this subsection we establish the convergence properties of the (first-order) distributed Riemannian Staircase (Algorithm 4.1) described in Theorem 4.3.

If Algorithm 4.1 terminates finitely (case (i)) then there is nothing to prove, so henceforward let us assume that Algorithm 4.1 generates an infinite sequence $\{X^{(r)}\}$ of factors in line 3. Our overall strategy is to exploit the correspondence between the critical points $X^{(r)}$ of Problem 4.3 (rank-restricted full SDP) and the critical points

$Y^{(r)}$ of Problem A.1 (rank-restricted rotation-only SDP) provided by Lemma A.1, together with the compactness of the feasible set of the SE-Sync relaxation Problem 4.2, to control the behavior of this sequence. To that end, define:

$$Z^{(r)} \triangleq (X^{(r)})^\top X^{(r)}, \qquad Z_{\mathrm{R}}^{(r)} \triangleq (Y^{(r)})^\top Y^{(r)}, \qquad \forall r \geq r_0, \tag{A.130}$$

where $Y^{(r)}$ is the first-order critical point of Problem A.1 obtained from $X^{(r)}$ as described by Lemma A.1. In addition, let $\Lambda^{(r)}$ and $\Lambda_{\mathrm{R}}^{(r)}$ denote the Lagrange multiplier matrices corresponding to $Z^{(r)}$ and $Z_{\mathrm{R}}^{(r)}$, respectively.

Since $\{Z_{\mathrm{R}}^{(r)}\}$ is an infinite sequence contained in the (compact) feasible set of Problem 4.2, it must contain a convergent subsequence $\{Z_{\mathrm{R}}^{(r_k)}\}$, with limit point $Z_{\mathrm{R}}^\star$. Since the Lagrange multiplier $\Lambda_{\mathrm{R}}^{(r_k)}$ is a continuous function of $Z_{\mathrm{R}}^{(r_k)}$ (see (A.23) and [14, eq. (107)]), it follows that $\{\Lambda_{\mathrm{R}}^{(r_k)}\}$ likewise converges to a limit $\Lambda_{\mathrm{R}}^\star$. By Lemma A.1(iii) (cf. (A.7)), it follows that the subsequence of Lagrange multipliers $\{\Lambda^{(r_k)}\}$ for the full (translation-explicit) problem also converges to a limit point $\Lambda^\star$, and consequently so does the sequence of certificate matrices $S^{(r_k)} \triangleq S(X^{(r_k)})$ computed in line 4 of Algorithm 4.1:

$$\lim_{k\to\infty} S^{(r_k)} = S^\star. \tag{A.131}$$

Now, let us consider two cases, corresponding to whether $S^\star$ is positive semidefinite.

Case 1: $S^\star \succeq 0$**.** It is easy to check that $S^{(r_k)}(X^{(r_k)})^\top = 0$ since $X^{(r_k)}$ is by definition a first-order critical point. This shows that $S^{(r_k)}$ has a zero eigenvalue. Using this property, together with the fact that the eigenvalues of a matrix S are continuous functions of S and that $S^\star \succeq 0$, it holds that $\lim_{k\to\infty} \lambda_{\min}\left(S(X^{(r_k)})\right) = 0$, thus proving (4.17).

To prove (4.16), note that by Lemma A.1(iv), the certificate matrix $S_{\mathrm{R}}^\star \triangleq Q_{\mathrm{R}} - \mathrm{SymBlockDiag}\,(Z_{\mathrm{R}}^\star Q_{\mathrm{R}})$ associated with the limit point $Z_{\mathrm{R}}^\star$ is likewise positive semidefinite, and therefore $Z_{\mathrm{R}}^\star$ is a minimizer of Problem 4.2 [14, Thm. 7]. Since $f(X^{(r)}) =$

$\langle Q_{\mathrm{R}}, {Y^{(r)}}^\top Y^{(r)} \rangle = \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^{(r)} \rangle$ for all $r \geq r_0$ (Lemma A.1(ii)), it follows that:

$$\lim_{k\to\infty} f(X^{(r_k)}) = \lim_{k\to\infty} \langle Q_{\mathrm{R}}, {Y^{(r_k)}}^\top Y^{(r_k)} \rangle = \lim_{k\to\infty} \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^{(r_k)} \rangle = \langle Q_{\mathrm{R}}, Z_{\mathrm{R}}^\star \rangle = f_{\mathrm{SDP}}^\star. \tag{A.132}$$

But then in fact we must have:

$$\lim_{r\to\infty} f(X^{(r)}) = f_{\mathrm{SDP}}^\star \tag{A.133}$$

since the sequence of objective values $\{f(X^{(r)})\}$ is *monotonically decreasing*. This establishes that Theorem 4.3(ii) holds for the case $S^\star \succeq 0$.

Case 2: $S^\star \not\succeq 0$**.** To finish the proof, we now show that in fact $S^\star \not\succeq 0$ cannot occur. To do so, suppose for contradiction that $\lambda_{\min}(S^\star) < 0$, and define $\mu \triangleq |\lambda_{\min}(S^\star)|$. Then there exists $k_1 > 0$ sufficiently large that $\lambda_{\min}(S^{(r_k)}) < -\mu/2$ for all $k > k_1$. Our aim is to show that this *upper* bound (away from 0) on the minimum eigenvalue of $S^{(r_k)}$ implies a *lower* bound $\delta > 0$ on the achievable decrease in the objective value each time the saddle escape procedure in line 10 of Algorithm 4.1 is invoked with any $k > k_1$; since this occurs infinitely many times, this would imply that the optimal value of Problem 4.1 is $-\infty$, a contradiction.

We note that the following analysis holds at any rank r_k of Algorithm 4.1 with $k > k_1$. For the ease of reading, however, we will suppress k in our notation and assume that k and r_k are clear from the context. Let X be the critical point generated by the distributed Riemannian Staircase at iteration rank r_k (Algorithm 4.1, line 3). Denote λ as the minimum eigenvalue of the corresponding certificate matrix (Algorithm 4.1, line 5), and $\dot{X} \in T_X \mathcal{M}_{\mathrm{PGO}}(r_k, n)$ as the corresponding second-order descent direction constructed (Algorithm 4.1, line 9). Recall that since $k > k_1$, we have $\lambda < -\mu/2$.

Consider the geodesic emanating from point X with initial velocity $\dot{X} = \dot{X}_+$ (see Theorem 4.6). With a slight abuse of notation, we denote this geodesic as $X(t)$ and

it may be parameterized as follows,[1]

$$X\colon \mathbb{R} \to \mathcal{M}_{\text{PGO}}(r_k, n) : t \mapsto \exp_X\left(t\dot{X}\right) \tag{A.134}$$

In addition, the cost function f of Problem 4.3 restricted to $X(t)$:

$$g\colon \mathbb{R} \to \mathbb{R}_{\geq 0} : t \mapsto f \circ X(t). \tag{A.135}$$

From the integral form of Taylor's Theorem, we have:

$$g(t) = g(0) + t\dot{g}(0) + \frac{t^2}{2}\ddot{g}(0) + \int_0^t \frac{\alpha^2}{2} g^{(3)}(\alpha)\, d\alpha. \tag{A.136}$$

Differentiating $g(t)$ and applying the chain rule, we arrive at,

$$\dot{g}(t) = \mathrm{D}f(X(t))[\dot{X}(t)] = \langle \operatorname{grad} f(X(t)), \dot{X}(t)\rangle, \tag{A.137}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product. Differentiating (A.137) again, we obtain,

$$\begin{aligned}
\ddot{g}(t) &= \frac{d}{dt}\big\langle \operatorname{grad} f(X(t)), \dot{X}(t)\big\rangle \\
&= \big\langle \frac{d}{dt}\left[\operatorname{grad} f(X(t))\right], \dot{X}(t)\big\rangle + \big\langle \operatorname{grad} f(X(t)), \ddot{X}(t)\big\rangle \\
&= \big\langle \operatorname{Hess} f(X(t))[\dot{X}(t)], \dot{X}(t)\big\rangle + \big\langle \operatorname{grad} f(X(t)), \ddot{X}(t)\big\rangle.
\end{aligned} \tag{A.138}$$

Equations (A.137)-(A.138) hold by applying standard results for embedded manifolds; see [18, Chapters 3 and 5] and [17]. Furthermore, since $X(t)$ is a geodesic, we may further show (see, *e.g.*, [17, Section 5.4]) that its extrinsic acceleration $\ddot{X}(t)$ is always orthogonal to the tangent space at $X(t)$. This means that the last term in (A.138) is identically zero since the Riemannian gradient at $X(t)$ belongs to the tangent space

[1] Note that this map is in fact well-defined on all of $\mathbb{R}$ because both the Stiefel manifold and $\mathbb{R}^n$ are *geodesically complete.*

at that point. Therefore, the second-order derivative further simplifies to,

$$\ddot{g}(t) = \left\langle \text{Hess}\, f(X(t))[\dot{X}(t)], \dot{X}(t) \right\rangle = 2\langle \dot{X}(t) S(X(t)), \dot{X}(t) \rangle. \tag{A.139}$$

The second equality makes use of (A.113). In particular, (A.137) and (A.139) show that $\dot{g}(0) = 0$ (since $X(0) = X$ is a critical point of f) and $\ddot{g}(0) = 2\lambda$ (since $\dot{X}(0) = \dot{X}$ is nothing but $\dot{X}_+$ in Theorem 4.6 and Remark 4.12). It follows from (A.136) that:

$$g(0) - g(t) = -\lambda t^2 - \int_0^t \frac{\alpha^2}{2} g^{(3)}(\alpha)\, d\alpha. \tag{A.140}$$

Thus, if we can exhibit scalars $\kappa, \tau > 0$ such that:

$$|g^{(3)}(t)| \leq \kappa \quad \forall t \in [0, \tau], \tag{A.141}$$

then (A.140) implies:

$$g(0) - g(t) \geq \underbrace{-\lambda t^2 - \frac{\kappa}{6} t^3}_{\delta(t)} \quad \forall t \in [0, \tau]. \tag{A.142}$$

Since:

$$\delta'(t) = -2\lambda t - \frac{\kappa}{2} t^2, \quad \delta''(t) = -2\lambda - \kappa t, \tag{A.143}$$

a straightforward calculation shows that the lower-bound $\delta(t)$ for the reduction in the objective value attained in (A.142) is maximized by the steplength:

$$t^* = -\frac{4\lambda}{\kappa}, \tag{A.144}$$

and that:

$$\delta'(t) > 0 \quad \forall t \in (0, t^*). \tag{A.145}$$

In consequence, by choosing a stepsize of:

$$t = \min\left\{ -\frac{4\lambda}{\kappa},\ \tau \right\} \geq \min\left\{ \frac{2\mu}{\kappa},\ \tau \right\} \triangleq \sigma \tag{A.146}$$

when performing the retraction in the saddle escape procedure, we can guarantee that the objective is decreased by at least:

$$\delta(t) \geq \delta(\sigma) = -\lambda\sigma^2 - \frac{\kappa}{6}\sigma^3 \geq \frac{\mu}{2}\sigma^2 - \frac{\kappa}{6}\sigma^3 \triangleq \delta > 0 \tag{A.147}$$

each time the saddle escape procedure is invoked, producing our desired contradiction.

It therefore suffices to exhibit fixed constants $\kappa, \tau > 0$ that will satisfy (A.141) for *all* invocations of the saddle escape procedure in line 10 with $k > k_1$. Before proceeding, we make one simplifying assumption that will remain in effect throughout the remainder of the proof: we assume without loss of generality that all iterates $X^{(r)}$ are translated so that the sum of the translational components $p_i^{(r)}$ is 0, as in Section A.2.4.[2] We now proceed in two stages: first we will identify an upper bound κ for the magnitude of $g^{(3)}(t)$ on the sublevel set of the initial value $f^{(0)}$ of f, and *then* a minimum distance τ from a given suboptimal critical point that we can retract along while still remaining inside this sublevel set.

Calculation of κ. Recall the following notation for the sublevel sets of the cost function f:

$$\mathcal{L}_f(r, n; \bar{f}) \triangleq \left\{ X \in \mathcal{M}_{\text{PGO}}(r, n) | f(X) \leq \bar{f} \right\}. \tag{A.148}$$

Our goal is to identify a fixed constant $\kappa > 0$ that upper bounds the magnitude of $g^{(3)}(t)$ within the *initial* sublevel set, *i.e.*,

$$|g^{(3)}(t)| \leq \kappa \quad \forall X(t) \in \mathcal{L}_f(r, n; f^{(r_0)}),\ r \geq r_0, \tag{A.149}$$

where $f^{(r_0)} \triangleq f(X^{(r_0)})$ is the *initial* value of the objective at the starting point $X^{(r_0)}$ supplied to Algorithm 4.1. From the second-order derivative (A.139), differentiate once again to obtain an explicit expression for $g^{(3)}(t)$. Here, we make use of the equality derived in (A.113) that relates the Riemannian Hessian and the certificate

[2]Note that this map leaves the objective f invariant, and is an isometry of both the ambient Euclidean space and $\mathcal{M}_{\text{PGO}}(r, n)$. Consequently, applying this transformation leaves all objective values and derivative norms invariant – we perform this transformation simply to achieve a parameterization that is more convenient for calculation.

matrix $S(X)$,

$$g^{(3)}(t) = \frac{d}{dt}\big\langle \operatorname{Hess} f(X(t))[\dot{X}(t)],\ \dot{X}(t)\big\rangle = 2\frac{d}{dt}\big\langle \dot{X}(t)S(X(t)),\ \dot{X}(t)\big\rangle. \tag{A.150}$$

Fully expanding the differentiation yields,

$$\begin{aligned} g^{(3)}(t) &= 2\big\langle \frac{d}{dt}[\dot{X}(t)S(X(t))],\ \dot{X}(t)\big\rangle + 2\big\langle \dot{X}(t)S(X(t)),\ \ddot{X}(t)\big\rangle \\ &= 2\big\langle \ddot{X}(t)S(X(t)) + \dot{X}(t)\frac{d}{dt}[S(X(t))],\ \dot{X}(t)\big\rangle + 2\big\langle \dot{X}(t)S(X(t)),\ \ddot{X}(t)\big\rangle \\ &= 2\big\langle \dot{X}(t)\frac{d}{dt}[S(X(t))],\ \dot{X}(t)\big\rangle + 4\big\langle \dot{X}(t)S(X(t)),\ \ddot{X}(t)\big\rangle \end{aligned} \tag{A.151}$$

where in the last step, we have used the symmetry of the certificate matrix $S(X(t))$ to combine terms. We proceed to bound the two inner products in (A.151) separately.

To bound the first inner product, we first use the Cauchy-Schwarz inequality followed by applying the submultiplicative property of the Frobenius norm:

$$\big|\big\langle \dot{X}(t)\frac{d}{dt}[S(X(t))], \dot{X}(t)\big\rangle\big| \le \big\|\dot{X}(t)\frac{d}{dt}[S(X(t))]\big\|_F \big\|\dot{X}(t)\big\|_F \le \big\|\frac{d}{dt}[S(X(t))]\big\|_F \big\|\dot{X}(t)\big\|_F^2. \tag{A.152}$$

By construction of the descent direction (4.45), the velocity vector at $t = 0$ satisfies $\|\dot{X}(0)\|_F = 1$. Since geodesics have constant speed, it thus holds that $\|\dot{X}(t)\|_F = 1$ for all t. Therefore, it suffices to bound the derivative of $S(X(t))$. Using the chain rule and repeatedly applying the submultiplicative property yields,

$$\begin{aligned} \left\|\frac{d}{dt}\left[S(X(t))\right]\right\|_F &= \left\|\mathrm{SymBlockDiag}_d^+\left(\dot{X}(t)^\mathsf{T}X(t)Q + X(t)^\mathsf{T}\dot{X}(t)Q\right)\right\|_F \\ &\le \big\|\mathrm{SymBlockDiag}_d^+\big\|_{Fop}\left\|\dot{X}(t)^\mathsf{T}X(t)Q + X(t)^\mathsf{T}\dot{X}(t)Q\right\|_F \\ &\le 2\big\|\mathrm{SymBlockDiag}_d^+\big\|_{Fop}\|Q\|_F\|X(t)\|_F\|\dot{X}(t)\|_F. \end{aligned} \tag{A.153}$$

Above, $\big\|\mathrm{SymBlockDiag}_d^+\big\|_{Fop}$ denotes the operator norm of the linear map $\mathrm{SymBlockDiag}_d^+$ with respect to the Frobenius norm. A closer examination of the definition of $\mathrm{SymBlockDiag}_d^+$

in (4.3) reveals that in fact:[3]

$$\|\text{SymBlockDiag}_d^+\|_{Fop} = 1. \tag{A.154}$$

Furthermore, since $X(t) \in \mathcal{L}_f(r, n; f^{(r_0)})$ and is assumed to be centered, applying the results in Appendix A.2.4 (Step 2) shows that,

$$\|X(t)\|_F = \sqrt{\|Y(t)\|_F^2 + \|p(t)\|_F^2} \leq \sqrt{dn + \frac{c_f}{\lambda_2(L)}}. \tag{A.155}$$

Finally, using that fact that the geodesic has constant speed $\|\dot{X}(t)\|_F = 1$, we establish the following constant bound for (A.153),

$$\left|\left\langle \dot{X}(t)\frac{d}{dt}[S(X(t))],\ \dot{X}(t)\right\rangle\right| \leq 2\|Q\|_F\sqrt{dn + \frac{c_f}{\lambda_2(L)}}. \tag{A.156}$$

Now, let us return to (A.151) to bound the second inner product. Once again, we start by invoking the Cauchy-Schwarz inequality and the submultiplicative property of the Frobenius norm,

$$\left|\left\langle \dot{X}(t)S(X(t)),\ \ddot{X}(t)\right\rangle\right| \leq \left\|\dot{X}(t)\right\|_F \|S(X(t))\|_F \left\|\ddot{X}(t)\right\|_F = \|S(X(t))\|_F \left\|\ddot{X}(t)\right\|_F. \tag{A.157}$$

Since $X(t) \in \mathcal{L}_f(r, n; f^{(r_0)})$, applying the result of Appendix A.2.4 (Step 2) shows that there exists a constant $c_s > 0$ such that $\|S(X(t))\|_F \leq c_s$. Next, we obtain an upper bound on the extrinsic acceleration. To this end, we note that in order to bound $\|\ddot{X}(t)\|_F$, it suffices to bound the extrinsic acceleration of each Stiefel element $\|\ddot{Y}_i(t)\|_F$ (since translations have zero extrinsic acceleration). Here, we use a result that gives an explicit characterization of $\ddot{Y}_i(t)$ along a geodesic on the Stiefel manifold

[3]This conclusion follows from the observation that SymBlockDiag_d^+ sets off-diagonal block elements of its argument to zero, and extracts the symmetric parts of the on-diagonal blocks; consequently SymBlockDiag_d^+ cannot increase the Frobenius norm of its argument, so that $\|\text{SymBlockDiag}_d^+\|_{Fop} \leq 1$. Equality follows from the fact that SymBlockDiag_d^+ *fixes* any symmetric block-diagonal matrix with the sparsity pattern appearing in (4.3).

[215, Eq. (2.7)]:

$$\ddot{Y}_i(t) + Y_i(t)[\dot{Y}_i(t)^\top \dot{Y}_i(t)] = 0. \tag{A.158}$$

From the above characterization, it holds that,

$$\|\ddot{Y}_i(t)\|_F \leq \|Y_i(t)\|_F \|\dot{Y}_i(t)\|_F^2 \leq \|Y_i(t)\|_F = \sqrt{d}, \tag{A.159}$$

where in the second inequality, we use the fact that $\|\dot{Y}_i(t)\|_F \leq \|\dot{X}(t)\|_F = 1$. Therefore, $\|\ddot{X}(t)\|_F$ is upper bounded as follows,

$$\|\ddot{X}(t)\|_F^2 = \sum_{i=1}^{n} \|\ddot{Y}_i(t)\|_F^2 \leq dn \implies \|\ddot{X}(t)\|_F \leq \sqrt{dn}. \tag{A.160}$$

Combining these results yields a constant bound for the second inner product (A.157):

$$\left|\left\langle \dot{X}(t)S(X(t)),\ \ddot{X}(t)\right\rangle\right| \leq c_s\sqrt{dn}. \tag{A.161}$$

Finally, using the upper bounds (A.156) and (A.161) in (A.151) yields the final fixed upper bound κ,

$$\begin{aligned} |g^{(3)}(t)| &= 2\left|\left\langle \dot{X}(t)\frac{d}{dt}[S(X(t))],\ \dot{X}(t)\right\rangle\right| + 4\left|\left\langle \dot{X}(t)S(X(t)),\ \ddot{X}(t)\right\rangle\right| \\ &\leq 4\|Q\|_F\sqrt{dn + \frac{c_f}{\lambda_2(L)}} + 4c_s\sqrt{dn} \\ &\triangleq \kappa. \end{aligned} \tag{A.162}$$

Calculation of τ: Now we derive a $\tau > 0$ such that $X(t) \in \mathcal{L}_f(r_k, n; f^{(r_0)})$ for all $t \in [0, \tau]$ and $k \geq k_1$. Our approach is based upon deriving a simple upper bound for the magnitude of the change in objective value between two points X_1 and X_2. To that end, consider:

$$\begin{aligned} |f(X_1) - f(X_2)| &= \left|\mathrm{tr}(QX_1^\top X_1) - \mathrm{tr}(QX_2^\top X_2)\right| \\ &= \left|\mathrm{tr}\left[Q\left(X_1^\top X_1 - X_2^\top X_2\right)\right]\right| \\ &\leq \|Q\|_F \left\|X_1^\top X_1 - X_2^\top X_2\right\|_F. \end{aligned} \tag{A.163}$$

Defining:

$$\Delta \triangleq X_2 - X_1, \tag{A.164}$$

we may substitute $X_2 = X_1 + \Delta$ in (A.163) to obtain:

$$\begin{aligned} |f(X_1) - f(X_2)| &\leq \|Q\|_F \left\| X_1^\top X_1 - (X_1 + \Delta)^\top (X_1 + \Delta) \right\|_F \\ &= \|Q\|_F \left\| X_1^\top \Delta + \Delta^\top X_1 + \Delta^\top \Delta \right\|_F \\ &\leq \|Q\|_F \left(2\|X_1\|_F \|\Delta\|_F + \|\Delta\|_F^2 \right). \end{aligned} \tag{A.165}$$

We will use inequality (A.165) to derive a *lower* bound τ on the admissible steplength t of the retraction applied at a critical point $X^{(r_k)}$ while still remaining inside the original sublevel set $\mathcal{L}_f(r_k, n; f^{(r_0)})$. Given the fact that $f^{(r_k)} < f^{(r_0)}$ and since, by definition, $X(t) \in \mathcal{L}_f(r_k, n; f^{(r_0)})$ iff $f(X(t)) \leq f^{(r_0)}$, it suffices to ensure that:

$$|f^{(r_k)} - f(X(t))| \leq f^{(r_0)} - f^{(r_k)}. \tag{A.166}$$

Applying inequality (A.165) with $X_1 = X^{(r_k)}$, $X_2 = X(t)$ and $\Delta = X(t) - X^{(r_k)}$,

$$|f^{(r_k)} - f(X(t))| \leq \|Q\|_F \left(2 \left\| X^{(r_k)} \right\|_F \|\Delta\|_F + \|\Delta\|_F^2 \right). \tag{A.167}$$

From (A.166) and (A.167), for $X(t)$ to remain in the sublevel set, it suffices to ensure that,

$$\|Q\|_F \left(2 \left\| X^{(r_k)} \right\|_F \|\Delta\|_F + \|\Delta\|_F^2 \right) \leq f^{(r_0)} - f^{(r_k)}. \tag{A.168}$$

After rearranging and simplification, we can arrive at the following equivalent condition:

$$\|X(t) - X^{(r_k)}\|_F = \|\Delta\|_F \leq \sqrt{\|X^{(r_k)}\|_F^2 + \frac{f^{(r_0)} - f^{(r_k)}}{\|Q\|_F}} - \left\| X^{(r_k)} \right\|_F. \tag{A.169}$$

It is worth noting that each invocation of escaping procedure strictly reduces the objective, and thus here $f^{(r_0)} > f^{(r_k)}$. We identify a constant lower bound for the right-hand side of (A.169). To this end, let us view the right-hand side as a function

of $\|X^{(r_k)}\|_F$. It can be straightforwardly verified that this function is *nonincreasing*. Since by (A.155) we have $\|X^{(r_k)}\|_F \leq \sqrt{dn + \frac{c_f}{\lambda_2(L)}}$, it follows that the right-hand side of (A.169) is always lower bounded the following constant,

$$\tau \triangleq \sqrt{dn + \frac{c_f}{\lambda_2(L)} + \frac{f^{(r_0)} - f^{(r_{k_1})}}{\|Q\|_F}} - \sqrt{dn + \frac{c_f}{\lambda_2(L)}}. \tag{A.170}$$

Recalling (A.169), we have shown that $X(t)$ remains in the sublevel set if the following holds:

$$\|X(t) - X^{(r_k)}\|_F \leq \tau. \tag{A.171}$$

Note that the above bound is expressed in terms of the chordal distance between $X(t)$ and $X^{(r_k)}$. To complete the proof, we note that the geodesic distance $d\left(X^{(r_k)}, X(t)\right) = t$ is always at least as large as the chordal distance. This means that for all $t \in [0, \tau]$, we have:

$$\|X(t) - X^{(r_k)}\|_F \leq d\left(X^{(r_k)}, X(t)\right) = t \leq \tau, \tag{A.172}$$

and thus $X(t) \in \mathcal{L}_f(r_k, n; f^{(r_0)})$, as claimed.

Appendix B

Supplemental Materials for Chapter 5

B.1 Proof of Lemma 5.2

Proof of Lemma 5.2. Suppose that at iteration k, robot i_k is selected for update. Recall that $x^k = \begin{bmatrix} x_1^k & x_2^k & \dots & x_n^k \end{bmatrix} \in \mathcal{M}$, where $\mathcal{M}$ is the product manifold $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 \times \dots \mathcal{M}_n$ (Section 5.4.1). For all $k \in \mathbb{N}$, define the aggregate tangent vector $\eta^k \in T_{x^k}\mathcal{M}$ as,

$$\eta_i^k \triangleq \begin{cases} \eta_{i_k}^k, & \text{if } i = i_k, \\ 0, & \text{otherwise.} \end{cases} \tag{B.1}$$

By definition of η^k, the update step in Algorithm 5.1 (line 6-7) is equivalent to,

$$x^{k+1} = \mathrm{Retr}_{x^k}(-\gamma \eta^k). \tag{B.2}$$

By Lemma 5.1, f has Lipschitz-type gradient for pullbacks. Therefore,

$$\begin{aligned} f(x^{k+1}) - f(x^k) &\leq -\gamma \langle \mathrm{grad}\, f(x^k), \eta^k \rangle + \frac{L\gamma^2}{2} \left\| \eta^k \right\|^2 \\ &= -\gamma \langle \mathrm{grad}_{i_k} f(x^k), \eta_{i_k}^k \rangle + \frac{L\gamma^2}{2} \left\| \eta_{i_k}^k \right\|^2. \end{aligned} \tag{B.3}$$

Using the equality $\langle \eta_1, \eta_2 \rangle = \frac{1}{2}[\|\eta_1\|^2 + \|\eta_2\|^2 - \|\eta_1 - \eta_2\|^2]$, we can convert the inner product that appears on the right-hand side of (B.3) into,

$$\langle \mathrm{grad}_{i_k} f(x^k), \eta_{i_k}^k \rangle = \frac{1}{2}\Big[\big\|\mathrm{grad}_{i_k} f(x^k)\big\|^2 + \big\|\eta_{i_k}^k\big\|^2 - \big\|\mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k\big\|^2 \Big]. \quad \text{(B.4)}$$

Substitute (B.4) into (B.3). After collecting relevant terms, we have,

$$\begin{aligned} f(x^{k+1}) - f(x^k) &\leq -\gamma \langle \mathrm{grad}_{i_k} f(x^k), \eta_{i_k}^k \rangle + \frac{L\gamma^2}{2} \big\|\eta_{i_k}^k\big\|^2 \\ &= -\frac{\gamma}{2}\Big[\big\|\mathrm{grad}_{i_k} f(x^k)\big\|^2 + \big\|\eta_{i_k}^k\big\|^2 - \big\|\mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k\big\|^2 \Big] + \frac{L\gamma^2}{2} \big\|\eta_{i_k}^k\big\|^2 \\ &= -\frac{\gamma}{2} \big\|\mathrm{grad}_{i_k} f(x^k)\big\|^2 - \frac{\gamma - L\gamma^2}{2} \big\|\eta_{i_k}^k\big\|^2 + \frac{\gamma}{2} \big\|\mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k\big\|^2 . \end{aligned} \quad \text{(B.5)}$$

We proceed to bound the last term on the right-hand side of (B.5). Recall from (5.5) and (5.6) that $\eta_{i_k}^k$ is formed with stale gradients,

$$\eta_{i_k}^k = \mathrm{grad}_{i_k} h_{i_k}(x_{i_k}^k) + \sum_{j_k \in \mathrm{Nbr}(i_k)} \mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k - B(j_k)}), \quad \text{(B.6)}$$

where we abbreviate the notation by defining $e_k \triangleq (i_k, j_k) \in E_{\mathcal{R}}$. In contrast, the Riemannian gradient $\mathrm{grad}_{i_k} f(x^k)$ is by definition formed using up-to-date variables,

$$\mathrm{grad}_{i_k} f(x^k) = \mathrm{grad}_{i_k} h_{i_k}(x_{i_k}^k) + \sum_{j_k \in \mathrm{Nbr}(i_k)} \mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k). \quad \text{(B.7)}$$

Note that the only difference between (B.6) and (B.7) is that delayed information is used in (B.6). In order to form the last term on the right-hand side of (B.5), we subtract (B.6) from (B.7) and compute the norm distance. Subsequently, we use the

triangle inequality to obtain an upper bound on this norm distance,

$$\begin{aligned}\left\|\mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k\right\| &= \left\| \sum_{e_k=(i_k,j_k)\in E_{\mathcal{R}}} \left[\mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right] \right\| \\ &\leq \sum_{e_k=(i_k,j_k)\in E_{\mathcal{R}}} \underbrace{\left\| \mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right\|}_{\epsilon(i_k,j_k)}.\end{aligned} \tag{B.8}$$

Next, we proceed to bound each $\epsilon(i_k, j_k)$ term. To do so, we use the fact that for a real-valued function f defined over a matrix submanifold $\mathcal{M} \subseteq \mathcal{E}$, its Riemannian gradient is obtained as the orthogonal projection of the Euclidean gradient onto the tangent space (see [17, Section. 3.6.1]),

$$\mathrm{grad}\, f(x) = \mathrm{Proj}_{T_x\mathcal{M}} \nabla f(x). \tag{B.9}$$

Substituting (B.9) into the right-hand side of (B.8), it holds that,

$$\begin{aligned}\epsilon(i_k, j_k) &= \left\| \mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \mathrm{grad}_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right\| \\ &= \left\| \mathrm{Proj}_{T_{x_{i_k}^k}} \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \mathrm{Proj}_{T_{x_{i_k}^k}} \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right\|.\end{aligned} \tag{B.10}$$

Furthermore, since the tangent space is identified as a linear subspace of the ambient space $\mathcal{E}$ [17, Section 3.5.7], the orthogonal projection operation is a *non-expansive* mapping, *i.e.*,

$$\begin{aligned}\epsilon(i_k, j_k) &= \left\| \mathrm{Proj}_{T_{x_{i_k}^k}} \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \mathrm{Proj}_{T_{x_{i_k}^k}} \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right\| \\ &\leq \left\| \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right\|.\end{aligned} \tag{B.11}$$

Since the norm distance with respect to i_k is no greater than the overall norm distance, we furthermore have,

$$\begin{aligned}\epsilon(i_k, j_k) &\leq \left\| \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \nabla_{i_k} f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right\| \\ &\leq \left\| \nabla f_{e_k}(x_{i_k}^k, x_{j_k}^k) - \nabla f_{e_k}(x_{i_k}^k, x_{j_k}^{k-B(j_k)}) \right\|.\end{aligned} \tag{B.12}$$

In (5.1), the Euclidean gradient of each cost function f_{e_k} is Lipschitz continuous. Furthermore, it is straightforward to show that the Lipschitz constant of f_{e_k} is always less than or equal to the Lipschitz constant of the overall cost function f. Denote the latter as $C > 0$. By definition, we thus have,

$$\epsilon(i_k, j_k) \leq C \left\| x_{j_k}^k - x_{j_k}^{k-B(j_k)} \right\|. \tag{B.13}$$

In addition, by Lemma A.3 in the appendix of Chapter 4, we know that Riemannian version of the Lipschitz constant L that appears in Lemma 5.1 is always greater than or equal to the Euclidean Lipschitz constant C.[1] Thus,

$$\epsilon(i_k, j_k) \leq L \left\| x_{j_k}^k - x_{j_k}^{k-B(j_k)} \right\|. \tag{B.14}$$

We proceed by bounding the norm on the right-hand side of (B.14). Writing the subtraction as a telescoping sum and invoking triangle inequality, we first obtain,

$$\left\| x_{j_k}^k - x_{j_k}^{k-B(j_k)} \right\| = \left\| \sum_{k'=k-B(j_k)}^{k-1} \left(x_{j_k}^{k'+1} - x_{j_k}^{k'} \right) \right\| \leq \sum_{k'=k-B(j_k)}^{k-1} \left\| x_{j_k}^{k'+1} - x_{j_k}^{k'} \right\|. \tag{B.15}$$

Recall that for all j_k and iterations k', the next iterate $x_{j_k}^{k'+1}$ is obtained from $x_{j_k}^{k'}$ via the following update,

$$x_{j_k}^{k'+1} = \mathrm{Retr}_{x_{j_k}^{k'}}(-\gamma \eta_{j_k}^{k'}). \tag{B.16}$$

Furthermore, in Lemma A.3 (specifically, (A.100)), it is shown that for each manifold $\mathcal{M}_i$, there exists a corresponding constant $\alpha_i > 0$ such that the Euclidean distance from the initial point to the new point after retraction is always bounded by the following quantity,

$$\left\| \mathrm{Retr}_{x_i}(\eta_i) - x_i \right\| \leq \alpha_i \left\| \eta_i \right\|, \ \forall x \in \mathcal{M}, \ \forall \eta_i \in T_{x_i}\mathcal{M}. \tag{B.17}$$

[1] Note that Lemma A.3 uses different notations for the Lipschitz constants. In particular, the Riemannian and Euclidean Lipschitz constants are denoted as c_g and c_l, respectively.

Equation (B.17) thus provides a way to bound the term on the right-hand side of (B.15),

$$\left\|x_{j_k}^k - x_{j_k}^{k-B(j_k)}\right\| \leq \sum_{k'=k-B(j_k)}^{k-1} \left\|\mathrm{Retr}_{x_{j_k}^{k'}}(-\gamma\eta_{j_k}^{k'}) - x_{j_k}^{k'}\right\| \leq \sum_{k'=k-B(j_k)}^{k-1} \alpha_{j_k} \left\|\gamma\eta_{j_k}^{k'}\right\|. \tag{B.18}$$

To remove the dependency on α_{j_k}, let $\alpha \triangleq \max_{i\in\mathcal{R}} \alpha_i$. We thus have,

$$\left\|x_{j_k}^k - x_{j_k}^{k-B(j_k)}\right\| \leq \alpha\gamma \sum_{k'=k-B(j_k)}^{k-1} \left\|\eta_{j_k}^{k'}\right\|. \tag{B.19}$$

We can further more use the bounded delay assumption (Assumption 5.1) to replace $B(j_k)$ with B,

$$\left\|x_{j_k}^k - x_{j_k}^{k-B(j_k)}\right\| \leq \alpha\gamma \sum_{k'=k-B(j_k)}^{k-1} \left\|\eta_{j_k}^{k'}\right\| \leq \alpha\gamma \sum_{k'=k-B}^{k-1} \left\|\eta_{j_k}^{k'}\right\|. \tag{B.20}$$

Substituting (B.20) into (B.14), we have,

$$\epsilon(i_k, j_k) \leq \alpha\gamma L \sum_{k'=k-B}^{k-1} \left\|\eta_{j_k}^{k'}\right\|. \tag{B.21}$$

Substituting (B.21) into (B.8), we then have,

$$\left\|\mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k\right\| \leq \sum_{j_k\in\mathrm{Nbr}(i_k)} \epsilon(i_k, j_k) \leq \alpha\gamma L \sum_{j_k\in\mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\|\eta_{j_k}^{k'}\right\|. \tag{B.22}$$

Squaring both sides of (B.22), we obtain,

$$\begin{aligned}\left\|\mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k\right\|^2 &\leq \left(\alpha\gamma L \sum_{j_k\in\mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\|\eta_{j_k}^{k'}\right\|\right)^2 \\ &= \alpha^2\gamma^2 L^2 \left(\sum_{j_k\in\mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\|\eta_{j_k}^{k'}\right\|\right)^2.\end{aligned} \tag{B.23}$$

Recall that the sum of squares inequality states that $(\sum_{i=1}^{n} a_i)^2 \leq n \sum_{i=1}^{n} a_i^2$. This gives an upper bound on the squared term in (B.23),

$$\begin{aligned} \left\| \mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k \right\|^2 &\leq \alpha^2 \gamma^2 L^2 B \Delta_{i_k} \sum_{j_k \in \mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\| \eta_{j_k}^{k'} \right\|^2 \\ &\leq \alpha^2 \gamma^2 L^2 B \Delta \sum_{j_k \in \mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\| \eta_{j_k}^{k'} \right\|^2, \end{aligned} \tag{B.24}$$

where $\Delta_{i_k} \leq \Delta$ is robot i_k's degree in the robot-level graph $G_{\mathcal{R}}$. Finally, substituting (B.24) in (B.5) concludes the proof,

$$\begin{aligned} & f(x^{k+1}) - f(x^k) \\ \leq & -\frac{\gamma}{2} \left\| \mathrm{grad}_{i_k} f(x^k) \right\|^2 - \frac{\gamma - L\gamma^2}{2} \left\| \eta_{i_k}^k \right\|^2 + \frac{\gamma}{2} \left\| \mathrm{grad}_{i_k} f(x^k) - \eta_{i_k}^k \right\|^2 \\ \leq & -\frac{\gamma}{2} \left\| \mathrm{grad}_{i_k} f(x^k) \right\|^2 - \frac{\gamma - L\gamma^2}{2} \left\| \eta_{i_k}^k \right\|^2 + \frac{\alpha^2 \gamma^3 L^2 B \Delta}{2} \sum_{j_k \in \mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\| \eta_{j_k}^{k'} \right\|^2. \end{aligned} \tag{B.25}$$

□

B.2 Proof of Theorem 5.1

Proof. Since $f^\star$ is a global lower bound on f, we can obtain the following inequality,

$$f^\star - f(x^0) \leq \mathbb{E}_{i_{0:K-1}} \Big[f(x^K) \Big] - f(x^0) = \mathbb{E}_{i_{0:K-1}} \Big[\sum_{k=0}^{K-1} (f(x^{k+1}) - f(x^k)) \Big], \tag{B.26}$$

where the right-hand side rewrites the middle term as a telescoping sum. Using the linearity of expectation, we obtain,

$$f^\star - f(x^0) \leq \mathbb{E}_{i_{0:K-1}} \Big[\sum_{k=0}^{K-1} (f(x^{k+1}) - f(x^k)) \Big] = \sum_{k=0}^{K-1} \mathbb{E}_{i_{0:k}} \Big[f(x^{k+1}) - f(x^k) \Big]. \tag{B.27}$$

For each expectation term, applying the law of total expectation yields,

$$f^\star - f(x^0) \leq \sum_{k=0}^{K-1} \mathbb{E}_{i_{0:k-1}} \Big[\mathbb{E}_{i_k|i_{0:k-1}} [f(x^{k+1}) - f(x^k)] \Big]. \tag{B.28}$$

Next, recall that Lemma 5.2 already gives an upper bound on the innermost term of (B.28). Substituting this upper bound into (B.28) gives,

$$\begin{aligned} f^\star - f(x^0) \leq \sum_{k=0}^{K-1} \mathbb{E}_{i_{0:k-1}} \Bigg[\mathbb{E}_{i_k|i_{0:k-1}} \Bigg[& -\frac{\gamma}{2} \left\| \mathrm{grad}_{i_k} f(x^k) \right\|^2 - \frac{\gamma - L\gamma^2}{2} \left\| \eta_{i_k}^k \right\|^2 \\ & + \frac{\Delta B \alpha^2 L^2 \gamma^3}{2} \sum_{j_k \in \mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\| \eta_{j_k}^{k'} \right\|^2 \Bigg] \Bigg]. \end{aligned} \tag{B.29}$$

Next, we simplify individual terms on the right-hand side of (B.29). We start with the first conditional expectation term. Using the definition of conditional expectation,

$$\mathbb{E}_{i_k|i_{0:k-1}} \Big[-\frac{\gamma}{2} \left\| \mathrm{grad}_{i_k} f(x^k) \right\|^2 \Big] = -\frac{\gamma}{2} \sum_{i=1}^{n} P(i_k = i | i_{0:k-1}) \left\| \mathrm{grad}_i f(x^k) \right\|^2. \tag{B.30}$$

Recall that $\{i_k\}$ are i.i.d. random variables uniformly distributed over 1 to n (Section 5.4.1). Setting $P(i_k = i|i_{0:k-1}) = 1/n$ thus gives,

$$\mathbb{E}_{i_k|i_{0:k-1}} \Big[-\frac{\gamma}{2} \left\| \mathrm{grad}_{i_k} f(x^k) \right\|^2 \Big] = -\frac{\gamma}{2} \sum_{i=1}^{n} \frac{1}{n} \left\| \mathrm{grad}_i f(x^k) \right\|^2 = -\frac{\gamma}{2n} \left\| \mathrm{grad}\, f(x^k) \right\|^2. \tag{B.31}$$

Similarly, for the third conditional expectation in (B.29), we note that,

$$\mathbb{E}_{i_k|i_{0:k-1}} \Big[\sum_{j_k \in \mathrm{Nbr}(i_k)} \sum_{k'=k-B}^{k-1} \left\| \eta_{j_k}^{k'} \right\|^2 \Big] = \sum_{i=1}^{n} \frac{1}{n} \sum_{j \in \mathrm{Nbr}(i)} \sum_{k'=k-B}^{k-1} \left\| \eta_j^{k'} \right\|^2. \tag{B.32}$$

In equation (B.32), exchange the order of summations and collect relevant terms,

$$\sum_{i=1}^{n} \frac{1}{n} \sum_{j \in \mathrm{Nbr}(i)} \sum_{k'=k-B}^{k-1} \left\| \eta_j^{k'} \right\|^2 = \frac{1}{n} \sum_{k'=k-B}^{k-1} \sum_{i=1}^{n} \sum_{j \in \mathrm{Nbr}(i)} \left\| \eta_j^{k'} \right\|^2 = \frac{2}{n} \sum_{k'=k-B}^{k-1} \left\| \eta^{k'} \right\|^2. \tag{B.33}$$

Using our simplified expressions for the first and third term on the right-hand side of (B.29), we obtain,

$$f^\star - f(x^0) \leq \sum_{k=0}^{K-1} \mathbb{E}_{i_{0:k-1}} \Bigg[-\frac{\gamma}{2n} \left\| \operatorname{grad} f(x^k) \right\|^2 - \mathbb{E}_{i_k | i_{0:k-1}} \left[\frac{\gamma - L\gamma^2}{2} \left\| \eta_{i_k}^k \right\|^2 \right] + \frac{\Delta B \alpha^2 L^2 \gamma^3}{n} \sum_{k'=k-B}^{k-1} \left\| \eta^{k'} \right\|^2 \Bigg]. \tag{B.34}$$

Next, using the independence relations and the linearity of expectation, we obtain,

$$f^\star - f(x^0) \leq \mathbb{E}_{i_{0:K-1}} \sum_{k=0}^{K-1} \Bigg[-\frac{\gamma}{2n} \left\| \operatorname{grad} f(x^k) \right\|^2 - \frac{\gamma - L\gamma^2}{2} \left\| \eta^k \right\|^2 + \frac{\Delta B \alpha^2 L^2 \gamma^3}{n} \sum_{k'=k-B}^{k-1} \left\| \eta^{k'} \right\|^2 \Bigg]. \tag{B.35}$$

At this point, our bound still involves the squared norms of update vectors from earlier iterations (last term on the right-hand side). To simplify the bound further, note that,

$$\sum_{k=0}^{K-1} \sum_{k'=k-B}^{k-1} \left\| \eta^{k'} \right\|^2 \leq B \sum_{k=0}^{K-1} \left\| \eta^k \right\|^2. \tag{B.36}$$

Using the above inequality in (B.35), we obtain,

$$f^\star - f(x^0) \leq \mathbb{E}_{i_{0:K-1}} \sum_{k=0}^{K-1} \Bigg[-\frac{\gamma}{2n} \left\| \operatorname{grad} f(x^k) \right\|^2 + \underbrace{\left(\frac{\Delta \alpha^2 B^2 L^2 \gamma^3}{n} + \frac{L\gamma^2 - \gamma}{2} \right)}_{A_1(\gamma)} \left\| \eta^k \right\|^2 \Bigg]. \tag{B.37}$$

We establish a sufficient condition on γ such that $A_1(\gamma)$ as a whole is nonpositive. Let us define $\rho \triangleq \Delta/n$. Consider the following factorization of $A_1(\gamma)$,

$$A_1(\gamma) = \frac{\gamma}{2} \underbrace{\left(2\rho\alpha^2 B^2 L^2 \gamma^2 + L\gamma - 1 \right)}_{A_2(\gamma)}. \tag{B.38}$$

Note that $A_2(\gamma)$ is the same as (5.11) in Theorem 5.1. For the moment, suppose that we can find $\gamma > 0$ such that $A_2(\gamma) \leq 0$. This implies that $A_1(\gamma) \leq 0$, and we can thus drop this term on the right-hand side of (B.37),

$$f^\star - f(x^0) \leq -\frac{\gamma}{2n} \sum_{k=0}^{K-1} \mathbb{E}_{i_{0:K-1}} \left[\left\| \operatorname{grad} f(x^k) \right\|^2 \right]. \tag{B.39}$$

Replacing the expected value at each iteration with the minimum across all iterations, we have,

$$f^\star - f(x^0) \leq -\frac{\gamma K}{2n} \min_{k \in \{0,\ldots,K-1\}} \mathbb{E}_{i_{0:K-1}} \left[\left\| \operatorname{grad} f(x^k) \right\|^2 \right]. \tag{B.40}$$

Finally, rearranging the last inequality yields,

$$\min_{k \in \{0,\ldots,K-1\}} \mathbb{E}_{i_{0:K-1}} \left[\left\| \operatorname{grad} f(x^k) \right\|^2 \right] \leq \frac{2n(f(x^0) - f^\star)}{\gamma K}. \tag{B.41}$$

Thus we have proved the main part of Theorem 5.1. To prove the expression (5.12), note that if $B = 0$ (*i.e.*, there is no delay), the condition $A_2(\gamma) \leq 0$ entails $L\gamma \leq 1$. In this case, we can thus set the upper bound $\bar{\gamma}$ to $1/L$. On the other hand, if $B > 0$, $A_2(\gamma)$ becomes a quadratic function of γ, and furthermore has the following positive root,

$$\bar{\gamma} = \frac{\sqrt{1 + 8\rho\alpha^2 B^2} - 1}{4\rho\alpha^2 B^2 L} > 0. \tag{B.42}$$

It is straightforward to verify that $A_2(\gamma) \leq 0$ for all $\gamma \in (0, \bar{\gamma}]$. Therefore, we have proved that the condition $A_2(\gamma) \leq 0$ is ensured by the following choice of $\bar{\gamma}$,

$$\bar{\gamma} = \begin{cases} \frac{\sqrt{1+8\rho\alpha^2 B^2}-1}{4\rho\alpha^2 B^2 L}, & B > 0, \\ 1/L, & B = 0. \end{cases} \tag{B.43}$$

In particular, under this choice, ASAPP converges globally to first-order critical points, with an associated convergence rate given in (B.41).

□

Appendix C

Supplemental Materials for Chapter 7

C.1 Proof of Lemma 7.1

Proof of Lemma 7.1. Recall from (7.6) that the global second-order approximation $\widehat{m}(u, v)$ involving both private vectors $u \in T_x\mathcal{X}$ and shared vector $v \in T_y\mathcal{Y}$ is defined as,

$$\widehat{m}(u,v) = f(x,y) + \left\langle \begin{bmatrix} g_x \\ g_y \end{bmatrix}, \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle + \frac{1}{2}\left\langle \begin{bmatrix} u \\ v \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle. \tag{C.1}$$

Setting the gradient of $\widehat{m}(u, v)$ with respect to u to zero yields,

$$\nabla_u \widehat{m}(u,v) = g_x + Au + Cv = 0 \implies u^\star(v) = -A^{-1}(Cv + g_x). \tag{C.2}$$

Recall the definition of A and C in (7.7). In particular, since A is a block-diagonal matrix, the i-th component of $u^\star(v)$ (corresponding to agent i) is given by,

$$u_i^\star(v) = -A_i^{-1}(C_i v + g_{ix}), \ \ \forall i \in [N]. \tag{C.3}$$

Next, substitute $u^\star(v)$ defined in (C.2) into (C.1). After collecting terms, we obtain,

$$\begin{aligned}\widehat{h}(v) &\triangleq \widehat{m}(u^\star(v), v) \\ &= f(x,y) - \frac{1}{2}\left\langle g_x, A^{-1} g_x \right\rangle + \left\langle \underbrace{g_y - C^\top A^{-1} g_x}_{w}, v \right\rangle + \frac{1}{2}\left\langle v, \underbrace{\left(B - C^\top A^{-1} C\right)}_{S} v \right\rangle.\end{aligned} \tag{C.4}$$

Consider the vector w as defined in (C.4). Note that the global Riemannian gradient with respect to y satisfies $g_y = \sum_{i=1}^N g_{iy}$ where $g_{iy} \triangleq \operatorname{grad}_y f_i(x_i, y)$. In addition, because of the block-diagonal structure of A in (7.7), $C^\top A^{-1} g_x = \sum_{i=1}^N C_i^\top A_i^{-1} g_{ix}$. Combining these results, we have that $w = \sum_{i=1}^N w_i$ where w_i is defined as in (7.10). Similarly, for the matrix S defined in (C.4), it can be readily verified that $S = \sum_{i=1}^N S_i$ where S_i is defined as in (7.11). □

C.2 Proof of Theorem 7.1

We start by reviewing several notations that are needed in this section. We use the superscript k to denote the value of a variable at iteration k of Algorithm 7.1. For example, g_x^k denotes the value of the Riemannian gradient g_x introduced in (7.6) at iteration k. Recall that $\|\cdot\|$ (without subscript) denotes the standard norm associated with the Riemannian metric. We also introduce an additional notation to simplify our presentation. Recall that $P^k : T_{y^k}\mathcal{Y} \to T_{y^k}\mathcal{Y}$ is the preconditioner used at iteration k to update the shared variable. In the following, we use $\langle \cdot, \cdot \rangle_k$ and $\|\cdot\|_k$ as shortcuts for $\langle \cdot, \cdot \rangle_{P^k}$ and $\|\cdot\|_{P^k}$, *i.e.*,

$$\langle v_1, v_2 \rangle_k \triangleq \langle v_1, v_2 \rangle_{P^k} = \left\langle v_1, P^k v_2 \right\rangle, \quad \forall v_1, v_2 \in T_{y^k}\mathcal{Y}, \tag{C.5}$$

$$\|v\|_k \triangleq \|v\|_{P^k} = \sqrt{\langle v, P^k v \rangle}, \quad \forall v \in T_{y^k}\mathcal{Y}. \tag{C.6}$$

In order to establish the convergence of Algorithm 7.1, we start by analyzing the change in the global objective (7.1a) after a single iteration (one step change). The following lemma provides an upper bound on the change in objective value.

Lemma C.1. *Under Assumption 7.1, each iteration of Algorithm 7.1 satisfies,*

$$f(x^{k+1},y^{k+1}) - f(x^k,y^k) \leq -\frac{1}{2L}\left\|g_x^k\right\|^2 - \frac{\gamma}{2}\left\|w^k\right\|_k^2 - \left(\frac{\gamma}{2} - \frac{\sigma_p\gamma^2}{2}\right)\left\|\widehat{w}^k\right\|_k^2 + \frac{\gamma}{2}\left\|\sum_{i=1}^N(\widehat{w}_i^k - w_i^k)\right\|_k^2. \tag{C.7}$$

Proof. Let $\widehat{m}^k(\cdot)$ and $\widehat{h}^k(\cdot)$ denote the second-order approximation in (7.6) and reduced second-order approximation in (7.9) at iteration k, respectively. Substitute $v^k = -\gamma P^k\widehat{w}^k$ into $\widehat{h}^k(\cdot)$:

$$\widehat{h}^k(v^k) = f(x^k,y^k) - \frac{1}{2}\left\langle g_x^k, (A^k)^{-1}g_x^k\right\rangle - \gamma\left\langle w^k, P^k\widehat{w}^k\right\rangle + \frac{\gamma^2}{2}\left\langle P^k\widehat{w}^k, S^kP^k\widehat{w}^k\right\rangle. \tag{C.8}$$

In (C.8), recall that $w^k = \sum_{i=1}^N w_i^k$ is the true global gradient computed using latest local gradients from all agents. On the other hand, $\widehat{w}^k = \sum_{i=1}^N \widehat{w}_i^k$ is the approximate global gradient computed using the lazily uploaded gradients from agents. Rearrange the third term in the right-hand side of (C.8) as,

$$\begin{aligned}
-\gamma\left\langle w^k, P^k\widehat{w}^k\right\rangle &= -\gamma\left\langle w^k, P^k(w^k + \widehat{w}^k - w^k)\right\rangle && \text{(C.9)}\\
&= -\gamma\left\|w^k\right\|_k^2 - \gamma\left\langle w^k, P^k(\widehat{w}^k - w^k)\right\rangle && \text{(C.10)}\\
&= -\gamma\left\|w^k\right\|_k^2 - \gamma\left\langle w^k, P^k\sum_{i=1}^N(\widehat{w}_i^k - w_i^k)\right\rangle && \text{(C.11)}\\
&= -\gamma\left\|w^k\right\|_k^2 + \left\langle -\sqrt{\gamma P^k}w^k, \sqrt{\gamma P^k}\sum_{i=1}^N(\widehat{w}_i^k - w_i^k)\right\rangle. && \text{(C.12)}
\end{aligned}$$

In (C.12), the matrix γP^k is positive definite, and we use $\sqrt{\gamma P^k}$ to denote its matrix square root. Recall the equality $\langle a,b\rangle = \frac{1}{2}\|a\|_2^2 + \frac{1}{2}\|b\|_2^2 - \frac{1}{2}\|a-b\|_2^2$. Applying this equality on the inner product term in (C.12), we obtain,

$$\left\langle -\sqrt{\gamma P^k}w^k, \sqrt{\gamma P^k}\sum_{i=1}^N(\widehat{w}_i^k - w_i^k)\right\rangle = \frac{\gamma}{2}\left\|w^k\right\|_k^2 + \frac{\gamma}{2}\left\|\sum_{i=1}^N(\widehat{w}_i^k - w_i^k)\right\|_k^2 - \frac{\gamma}{2}\left\|\widehat{w}^k\right\|_k^2. \tag{C.13}$$

Substitute (C.13) into (C.12), we obtain,

$$-\gamma \left\langle w^k, P^k \widehat{w}^k \right\rangle = -\frac{\gamma}{2} \left\| w^k \right\|_k^2 - \frac{\gamma}{2} \left\| \widehat{w}^k \right\|_k^2 + \frac{\gamma}{2} \left\| \sum_{i=1}^{N} (\widehat{w}_i^k - w_i^k) \right\|_k^2 . \tag{C.14}$$

Now, let us focus on the last term in (C.8). Applying the Cauchy-Schwartz inequality with respect to the norm induced by P^k, it holds that,

$$\begin{aligned} \frac{\gamma^2}{2} \left\langle P^k \widehat{w}^k, S^k P^k \widehat{w}^k \right\rangle &= \frac{\gamma^2}{2} \left\langle \widehat{w}^k, S^k P^k \widehat{w}^k \right\rangle_k && \text{(C.15)} \\ &\leq \frac{\gamma^2}{2} \left\| \widehat{w}^k \right\|_k \left\| S^k P^k \widehat{w}^k \right\|_k && \text{(C.16)} \\ &\leq \frac{\sigma_p \gamma^2}{2} \left\| \widehat{w}^k \right\|_k^2 . && \text{(C.17)} \end{aligned}$$

The last inequality (C.17) holds because of assumption (A3). Finally, substitute (C.14) and (C.17) into (C.8), we obtain that,

$$\begin{aligned} \widehat{h}^k(v^k) \leq & f(x^k, y^k) - \frac{1}{2} \left\langle g_x^k, (A^k)^{-1} g_x^k \right\rangle - \frac{\gamma}{2} \left\| w^k \right\|_k^2 - \left(\frac{\gamma}{2} - \frac{\sigma_p \gamma^2}{2} \right) \left\| \widehat{w}^k \right\|_k^2 \\ & + \frac{\gamma}{2} \left\| \sum_{i=1}^{N} (\widehat{w}_i^k - w_i^k) \right\|_k^2 . \end{aligned} \tag{C.18}$$

This further implies that,

$$\begin{aligned} \widehat{h}^k(v^k) \leq & f(x^k, y^k) - \frac{1}{2L} \left\| g_x^k \right\|^2 - \frac{\gamma}{2} \left\| w^k \right\|_k^2 - \left(\frac{\gamma}{2} - \frac{\sigma_p \gamma^2}{2} \right) \left\| \widehat{w}^k \right\|_k^2 \\ & + \frac{\gamma}{2} \left\| \sum_{i=1}^{N} (\widehat{w}_i^k - w_i^k) \right\|_k^2 , \end{aligned} \tag{C.19}$$

where the last inequality holds, because $A^k \preceq LI$ due to assumption (A2). To conclude the proof, note that (A1) and (A2) together imply that the model function $\widehat{m}^k$

is an upper bound on the current pullback function $\widehat{f}^k$,

$$\widehat{f}^k(u,v) \leq f(x^k,y^k) + \left\langle \begin{bmatrix} g_x^k \\ g_y^k \end{bmatrix}, \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle + \frac{c_g}{2} \left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\|^2 \tag{C.20}$$

$$\leq f(x^k,y^k) + \left\langle \begin{bmatrix} g_x^k \\ g_y^k \end{bmatrix}, \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle + \frac{1}{2} \left\langle \begin{bmatrix} u \\ v \end{bmatrix}, M^k \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle \tag{C.21}$$

$$\triangleq \widehat{m}^k(u,v). \tag{C.22}$$

Above, the first inequality holds due to the Lipschitz-type gradient conditions for pullback (A1), and the second inequality holds because $M^k \succeq c_g I$ (A2). The above inequality directly shows that,

$$f(x^{k+1},y^{k+1}) = \widehat{f}^k(u^k,v^k) \leq \widehat{m}^k(u^k,v^k) = \widehat{m}^k(u^\star(v^k),v^k) = \widehat{h}^k(v^k). \tag{C.23}$$

This concludes the proof. □

In Lemma C.1, the RHS of (C.7) bounds the absolute reduction in the global cost function after each iteration of Algorithm 7.1. However, due to the last term in (C.7) (which captures the error between the approximate gradient $\widehat{w}^k$ and true w^k), the RHS can in general be positive. This means that we cannot directly use (C.7) to show that Algorithm 7.1 decreases the global cost function at every iteration, *i.e.*, Algorithm 7.1 is not a descent method with respect to the cost function f. Fortunately, we can still show that Algorithm 7.1 is a descent method with respect to a *Lyapunov function*, which is sufficient for proving convergence. This proof technique is inspired by LAG [164]. Specifically, we define the Lyapunov function to be the sum of global objective and the squared norms of past approximate gradients,

$$V^k \triangleq f(x^k,y^k) + \sum_{d=1}^{\bar{d}} \beta_d \left\| \widehat{w}^{k-d} \right\|_{k-d}^2, \tag{C.24}$$

where $\{\beta_d \geq 0, d = 1,\ldots,\bar{d}\}$ are constants to be specified. Note that V^k combines the current cost function with weighted squared norms of past approximate reduced

gradients. Intuitively, these squared norms account for the approximation errors induced by lazy communication, and allows us to establish the convergence of the proposed method.

Lemma C.2 (Descent lemma)**.** *Under Assumption 7.1, there exist suitable choices of parameters $\gamma, \{\beta_d\}, \{\epsilon_d\}$, such that each iteration of Algorithm 7.1 satisfies,*

$$V^{k+1} - V^k \leq -\frac{1}{2L}\left\|g_x^k\right\|^2 - \alpha_0\left\|w^k\right\|_k^2 - \sum_{d=1}^{\bar{d}} \alpha_d \left\|\widehat{w}^{k-d}\right\|_{k-d}^2 \leq 0, \tag{C.25}$$

where $\alpha_0, \ldots, \alpha_{\bar{d}} > 0$ are fixed constants. In particular, the following provides a set of admissible conditions on the parameters such that (C.25) *holds,*

$$0 < \gamma < 1/\sigma_p, \tag{C.26}$$

$$\beta_1 = (\gamma - \sigma_p\gamma^2)/2, \tag{C.27}$$

$$\beta_d < \beta_{d-1} - \gamma\epsilon_{d-1}/2,\ d = 2, \ldots, \bar{d}, \tag{C.28}$$

$$\beta_{\bar{d}} > \gamma\epsilon_{\bar{d}}/2. \tag{C.29}$$

Proof. Consider the difference between the Lyapunov function between the current and next iterations,

$$V^{k+1} - V^k = f(x^{k+1}, y^{k+1}) - f(x^k, y^k) + \sum_{d=1}^{\bar{d}} \beta_d \left\|\widehat{w}^{k-d+1}\right\|_{k-d+1}^2 - \sum_{d=1}^{\bar{d}} \beta_d \left\|\widehat{w}^{k-d}\right\|_{k-d}^2. \tag{C.30}$$

Using Lemma C.1, we can obtain the following upper bound,

$$\begin{aligned} V^{k+1} - V^k \leq & -\frac{1}{2L}\left\|g_x^k\right\|^2 - \frac{\gamma}{2}\left\|w^k\right\|_k^2 - \left(\frac{\gamma}{2} - \frac{\sigma_p\gamma^2}{2}\right)\left\|\widehat{w}^k\right\|_k^2 + \frac{\gamma}{2}\left\|\sum_{i=1}^{N}(\widehat{w}_i^k - w_i^k)\right\|_k^2 \\ & + \sum_{d=1}^{\bar{d}} \beta_d \left\|\widehat{w}^{k-d+1}\right\|_{k-d+1}^2 - \sum_{d=1}^{\bar{d}} \beta_d \left\|\widehat{w}^{k-d}\right\|_{k-d}^2. \end{aligned} \tag{C.31}$$

Grouping terms that involve $\left\|\widehat{w}^k\right\|_k^2$ together, we obtain,

$$V^{k+1}-V^k \leq -\frac{1}{2L}\left\|g_x^k\right\|^2-\frac{\gamma}{2}\left\|w^k\right\|_k^2+\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)\left\|\widehat{w}^k\right\|_k^2+\frac{\gamma}{2}\left\|\sum_{i=1}^N(\widehat{w}_i^k-w_i^k)\right\|_k^2 + \sum_{d=2}^{\bar{d}}\beta_d\left\|\widehat{w}^{k-d+1}\right\|_{k-d+1}^2-\sum_{d=1}^{\bar{d}}\beta_d\left\|\widehat{w}^{k-d}\right\|_{k-d}^2. \tag{C.32}$$

Next, we obtain an upper bound for $\left\|\widehat{w}^k\right\|_k^2$. First, using the definition of the approximate gradient $\widehat{w}^k$, it holds that,

$$\left\|\widehat{w}^k\right\|_k^2=\left\|w^k+\sum_{i=1}^N(\widehat{w}_i^k-w_i^k)\right\|_k^2=\left\|\sqrt{P^k}w^k+\sqrt{P^k}\sum_{i=1}^N(\widehat{w}_i^k-w_i^k)\right\|^2. \tag{C.33}$$

Next, applying Young's inequality, we arrive at the following upper bound,

$$\left\|\widehat{w}^k\right\|_k^2\leq(1+\rho)\left\|\sqrt{P^k}w^k\right\|^2+(1+\rho^{-1})\left\|\sqrt{P^k}\sum_{i=1}^N(\widehat{w}_i^k-w_i^k)\right\|^2 \tag{C.34}$$

$$=(1+\rho)\left\|w^k\right\|_k^2+(1+\rho^{-1})\left\|\sum_{i=1}^N(\widehat{w}_i^k-w_i^k)\right\|_k^2, \tag{C.35}$$

where $\rho>0$ is any constant. Plug (C.35) into (C.32). After grouping terms, we arrive at,

$$V^{k+1}-V^k\leq-\frac{1}{2L}\left\|g_x^k\right\|^2+\left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho)-\frac{\gamma}{2}\right]\left\|w^k\right\|_k^2 + \left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho^{-1})+\frac{\gamma}{2}\right]\left\|\sum_{i=1}^N(\widehat{w}_i^k-w_i^k)\right\|_k^2 + \sum_{d=1}^{\bar{d}-1}(\beta_{d+1}-\beta_d)\left\|\widehat{w}^{k-d}\right\|_{k-d}^2-\beta_{\bar{d}}\left\|\widehat{w}^{k-\bar{d}}\right\|_{k-\bar{d}}^2. \tag{C.36}$$

Next, we obtain an upper bound for the second row of (C.36). Note that,

$$\left\|\sum_{i=1}^{N}(\widehat{w}_i^k - w_i^k)\right\|_k^2 = \left\|\sum_{i=1}^{N}\sqrt{P^k}(\widehat{w}_i^k - w_i^k)\right\|^2 \tag{C.37}$$

$$\leq N\sum_{i=1}^{N}\left\|\sqrt{P^k}(\widehat{w}_i^k - w_i^k)\right\|^2 \tag{C.38}$$

$$= N\sum_{i=1}^{N}\left\|\widehat{w}_i^k - w_i^k\right\|_k^2. \tag{C.39}$$

Recall the communication triggering condition (7.20). By definition, (7.20) guarantees that the approximation error for each block l is upper bounded as follows,

$$\left\|\widehat{w}_{il}^k - w_{il}^k\right\|_{P_l^k}^2 \leq \frac{1}{mN^2}\sum_{d=1}^{\bar{d}}\epsilon_d\left\|\widehat{w}^{k-d}\right\|_{k-d}^2. \tag{C.40}$$

Summing the above inequality over all m blocks, we can obtain an upper bound on the approximation error for the entire local gradient,

$$\left\|\widehat{w}_i^k - w_i^k\right\|_k^2 = \sum_{l=1}^{m}\left\|\widehat{w}_{il}^k - w_{il}^k\right\|_{P_l^k}^2 \leq \frac{1}{N^2}\sum_{d=1}^{\bar{d}}\epsilon_d\left\|\widehat{w}^{k-d}\right\|_{k-d}^2. \tag{C.41}$$

Substitute (C.41) into (C.39),

$$\left\|\sum_{i=1}^{N}(\widehat{w}_i^k - w_i^k)\right\|_k^2 \leq N\sum_{i=1}^{N}\left(\frac{1}{N^2}\sum_{d=1}^{\bar{d}}\epsilon_d\left\|\widehat{w}^{k-d}\right\|_{k-d}^2\right) \tag{C.42}$$

$$= \sum_{d=1}^{\bar{d}}\epsilon_d\left\|\widehat{w}^{k-d}\right\|_{k-d}^2. \tag{C.43}$$

Substitute (C.43) into the second row of (C.36),

$$\begin{aligned} V^{k+1}-V^k \leq & -\frac{1}{2L}\left\|g_x^k\right\|^2+\left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho)-\frac{\gamma}{2}\right]\left\|w^k\right\|_k^2 \\ & +\left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho^{-1})+\frac{\gamma}{2}\right]\sum_{d=1}^{\bar{d}}\epsilon_d\left\|\widehat{w}^{k-d}\right\|_{k-d}^2 \\ & +\sum_{d=1}^{\bar{d}-1}(\beta_{d+1}-\beta_d)\left\|\widehat{w}^{k-d}\right\|_{k-d}^2-\beta_{\bar{d}}\left\|\widehat{w}^{k-\bar{d}}\right\|_{k-\bar{d}}^2. \end{aligned} \tag{C.44}$$

After grouping terms in (C.44), we arrive at,

$$\begin{aligned} V^{k+1}-V^k \leq & -\frac{1}{2L}\left\|g_x^k\right\|^2-\left[\frac{\gamma}{2}-\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho)\right]\left\|w^k\right\|_k^2 \\ & -\sum_{d=1}^{\bar{d}-1}\left\{\beta_d-\beta_{d+1}-\epsilon_d\left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho^{-1})+\frac{\gamma}{2}\right]\right\}\left\|\widehat{w}^{k-d}\right\|_{k-d}^2 \\ & -\left\{\beta_{\bar{d}}-\epsilon_{\bar{d}}\left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho^{-1})+\frac{\gamma}{2}\right]\right\}\left\|\widehat{w}^{k-\bar{d}}\right\|_{k-\bar{d}}^2. \end{aligned} \tag{C.45}$$

Define the following constants that correspond to the coefficients in the above inequality,

$$\alpha_0 \triangleq \frac{\gamma}{2}-\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho), \tag{C.46}$$

$$\alpha_d \triangleq \beta_d-\beta_{d+1}-\epsilon_d\left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho^{-1})+\frac{\gamma}{2}\right],\ d=1,\ldots,\bar{d}-1, \tag{C.47}$$

$$\alpha_{\bar{d}} \triangleq \beta_{\bar{d}}-\epsilon_{\bar{d}}\left[\left(\beta_1-\frac{\gamma}{2}+\frac{\sigma_p\gamma^2}{2}\right)(1+\rho^{-1})+\frac{\gamma}{2}\right]. \tag{C.48}$$

For the Lyapunov function to be decreasing, it suffices to choose $\gamma, \{\epsilon_d\}, \{\beta_d\}$ such that $\alpha_d > 0$ for all $d = 0, 1, \ldots, \bar{d}$. To conclude the proof, we show that the conditions outlined in (C.26)-(C.29), which are inspired by similar conditions in [164], indeed satisfy this requirement. Let us assume that $\{\beta_d\}$ is a decreasing sequence, *i.e.*, $\beta_1 > \beta_2 > \ldots > \beta_{\bar{d}}$. This assumption makes intuitive sense, since it assigns larger weights to more recent gradients in the definition of the Lyapunov function (C.24). In addition, let us also assume that the stepsize satisfies $0 < \gamma < 1/\sigma_p$. Note the

similarity of this assumption with the condition $0 < \gamma < 1/L$ that is commonly used to ensure the convergence of gradient descent (*e.g.*, see [167]). Under these two simplifications, let us choose $\beta_1 = (\gamma - \sigma_p \gamma^2)/2 > 0$, so that $\beta_1 - \frac{\gamma}{2} + \frac{\sigma_p \gamma^2}{2} = 0$. Then, it can be verified that the following conditions ensure $\alpha_d > 0$ for all $d = 0, 1, \ldots, \bar{d}$:

$$\beta_d - \beta_{d+1} - \epsilon_d \gamma/2 > 0, \ d = 1, \ldots, \bar{d} - 1, \tag{C.49}$$

$$\beta_{\bar{d}} - \epsilon_{\bar{d}} \gamma/2 > 0. \tag{C.50}$$

We can verify that the above conditions are equivalent to (C.26)-(C.29). □

Remark C.1 (Intuitions behind parameter settings)**.** Before proceeding, let us provide more insights on the choice of algorithm parameters (C.26)-(C.29) outlined in Lemma C.2. For this purpose, let us focus on the special case when $\bar{d} = 1$, *i.e.*, only a single past gradient is used in the calculation of the communication triggering condition (7.20). In this case, it can be shown that the conditions (C.26)-(C.29) reduce to the following,

$$0 < \gamma < 1/\sigma_p, \tag{C.51}$$

$$\beta_1 = (\gamma - \sigma_p \gamma^2)/2, \tag{C.52}$$

$$\epsilon_1 < 2\beta_1/\gamma = 1 - \sigma_p \gamma. \tag{C.53}$$

In particular, the last inequality demonstrates the intuitive *trade-off* between the stepsize γ and the lazy communication threshold ϵ_1: *with smaller stepsize, we can tolerate larger approximation errors (and hence save more communication) at each iteration.*

With Lemma C.2, we are ready to prove Theorem 7.1 which is stated in Section 7.4 and is repeated below.

Theorem 7.1. Under Assumption 7.1 and the conditions in Lemma C.2, after K iterations, the iterates generated by Algorithm 7.1 satisfy,

$$\min_{k \in [K]} \left\| \operatorname{grad} f(x^k, y^k) \right\|^2 = O(1/K). \tag{C.54}$$

Proof. Using Lemma C.2 we know that,

$$\frac{1}{2L}\left\|g_x^k\right\|^2 + \alpha_0 \left\|w^k\right\|_k^2 \leq V^k - V^{k+1} \tag{C.55}$$

Furthermore, from Assumption (A3),

$$\frac{1}{2L}\left\|g_x^k\right\|^2 + \alpha_0\mu_p \left\|w^k\right\|^2 \leq V^k - V^{k+1} \tag{C.56}$$

Define $\alpha \triangleq \min(1/2L, \alpha_0\mu_p)$,

$$\alpha\left(\left\|g_x^k\right\|^2 + \left\|w^k\right\|^2\right) \leq V^k - V^{k+1} \tag{C.57}$$

A telescoping sum of (C.57) from $k = 1$ to $k = K$ yields,

$$\alpha\sum_{k=1}^{K}\left(\left\|g_x^k\right\|^2 + \left\|w^k\right\|^2\right) \leq V^1 - V^{K+1} \leq V^1 - f^\star. \tag{C.58}$$

Above, $f^\star$ denotes the global minimum of Problem 7.1. The second inequality holds, because by definition of the Lyapunov function (C.24) we have $V^k \geq f^\star$ for all k. Inequality (C.58) further implies that,

$$\min_{k\in[K]}\left(\left\|g_x^k\right\|^2 + \left\|w^k\right\|^2\right) \leq \frac{V^1 - f^\star}{\alpha K}. \tag{C.59}$$

To conclude the proof, we show that (C.59) implies (7.25) in Theorem 7.1. From now on, let k denote the iteration that minimizes the LHS of (C.59). In addition, define $\varepsilon \triangleq \sqrt{(V^1 - f^\star)/(\alpha K)}$. Then, (C.59) implies that we have both $\left\|g_x^k\right\| \leq \varepsilon$ and $\left\|w^k\right\| \leq \varepsilon$. Recall from (C.4) that $w = g_y - C^\top A^{-1} g_x$. From (A2) we have that $\|A^{-1}\| \leq \mu^{-1}$. Also, since the approximate Hessian M (7.6) is positive definite,

$$M = \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix} \succeq 0, \tag{C.60}$$

it holds that $C = A^{1/2} Z B^{1/2}$ where $\|Z\| \leq 1$ [216, Lemma 3.5.12]. Therefore we also

have that $\|C\| \leq L$. Applying these results together with the triangle inequality,

$$\left\|g_y^k\right\| \leq \left\|w^k\right\| + \left\|C^\top A^{-1} g_x^k\right\| \leq \left(1 + \frac{L}{\mu}\right) \varepsilon. \tag{C.61}$$

Lastly, for $g^k \triangleq \operatorname{grad} f(x^k, y^k)$, it holds that

$$\left\|g^k\right\|^2 = \left\|g_x^k\right\|^2 + \left\|g_y^k\right\|^2 \leq \left[1 + \left(1 + \frac{L}{\mu}\right)^2\right] \varepsilon^2 = \left[1 + \left(1 + \frac{L}{\mu}\right)^2\right] \frac{V^1 - f^\star}{\alpha K}. \tag{C.62}$$

The proof is completed. □

Appendix D

Supplemental Materials for Chapter 8

Table D.1: Summary of key notations used in this Chapter 8 (organized by sections).

Notation	Description	Reference
	Section 8.2	
$G = (\mathcal{V}, \mathcal{E})$	Multi-robot measurement graph with vertex (variable) set $\mathcal{V}$ and edge (measurement) set $\mathcal{E}$	
R_i	The ith rotation variable to be estimated in rotation averaging	(8.5)
$\widetilde{R}_{ij}$	Noisy relative rotation measurement in rotation averaging	(8.5)
$\varphi(\cdot, \cdot)$	Squared geodesic or chordal distance function between two rotations	(8.6a)-(8.6b)
t_i	The ith translation variable to be estimated in translation estimation	(8.7)
$\widehat{t}_{ij}$	Noisy relative translation measurement in translation estimation	(8.7)
$\widetilde{t}_{ij}$	Noisy relative translation measurement in PGO	(8.8)
κ_{ij}	Weight (precision) associated with the relative rotation measurement between vertex i and j	(8.5), (8.8)
τ_{ij}	Weight (precision) associated with the relative translation measurement between vertex i and j	(8.7), (8.8)
	Section 8.3	
p	$p \triangleq \dim \mathrm{SO}(d)$ is the intrinsic dimension of the rotation group $\mathrm{SO}(d)$	
$[R]$	The equivalent class corresponding to n rotations $R = (R_1, \ldots, R_n) \in \mathrm{SO}(d)^n$	(8.9)
v_i	$v_i \in \mathbb{R}^p$ is the correction vector (to be optimized) for rotation variable R_i	
v	$v \in \mathbb{R}^{pn}$ is formed by concatenating the v_i vectors of all n rotation variables	(8.10)
V	$V \in \mathbb{R}^{n \times p}$ is the matrix representation of v	(8.18)
$\mathcal{N}, \mathcal{H}$	Subspaces of $\mathbb{R}^{pn}$ corresponding to the vertical space and horizontal space in rotation averaging ($\mathcal{H} \triangleq \mathcal{V}^{\perp}$)	(8.11)
P_H	Orthogonal projection onto the horizontal space $\mathcal{H}$	(8.13)
$\overline{g}(R)$	$\overline{g}(R) \in \mathbb{R}^{pn}$ is the vector corresponding to the Riemannian gradient of rotation averaging in the total space	(8.12)
$\overline{H}(R)$	$\overline{H}(R) \in \mathcal{S}^{pn}$ is the matrix corresponding to the Riemannian Hessian of rotation averaging in the total space	(8.12)
$H(R)$	$H(R) \in \mathcal{S}^{pn}$ is the matrix corresponding to the Riemannian Hessian in the quotient space	(8.13)
w	$w : \mathcal{E} \to \mathbb{R}_{>0}$ is the edge weight function that appears in Theorem 8.1	(8.15)
δ	The approximation constant in Theorem 8.1 between $H(R)$ and the Laplacian $L(G; w) \otimes I_p$	(8.15)
μ_H	Lower bound of $H(R)$ in Corollary 8.1	(8.16)
L_H	Upper bound of $H(R)$ in Corollary 8.1	(8.16)
κ_H	Condition number of $H(R)$ as defined by $\kappa_H = L_H/\mu_H$	Cor. 8.1
$B(R)$	$B(R) \in \mathbb{R}^{n \times p}$ is the matrix representation of the negative Riemannian gradient $-\overline{g}(R)$	(8.18)
	Section 8.4	
$\mathcal{V}_\alpha$	The set of vertices (variables) of robot $\alpha \in [m]$, and $\mathcal{V}_\alpha = \mathcal{F}_\alpha \uplus \mathcal{C}_\alpha$	(8.22)
$\mathcal{F}_\alpha$	$\mathcal{F}_\alpha \subseteq \mathcal{V}_\alpha$ is the set of interior vertices of robot α that does not have inter-robot measurement	
$\mathcal{C}_\alpha$	$\mathcal{C}_\alpha \subseteq \mathcal{V}_\alpha$ is the set of separator vertices of robot α hat have inter-robot measurement	
$\mathcal{C}$	$\mathcal{C} = \mathcal{C}_1 \uplus \ldots \uplus \mathcal{C}_m$ is the union of separator vertices of all m robots	
$\mathcal{E}_\alpha$	The set of local edges (measurements) of robot $\alpha \in [m]$	(8.23)
$\mathcal{E}_c$	The set of inter-robot edges (measurements)	(8.23)
S_α	$S_\alpha \in \mathcal{S}_+^{|\mathcal{C}|}$ is the exact Schur complement of robot α's local graph G_α	
S	$S \in \mathcal{S}_+^{|\mathcal{C}|}$ is the exact Schur complement of the multi-robot measurement graph	(8.27)
$\widetilde{S}_\alpha$	The sparsified version of S_α robot α transmits to the server in Algorithm 8.2	Alg. 8.2, line 4
$\widetilde{S}$	The sparsified version of S computed by the server in Algorithm 8.2	Alg. 8.2, line 6
U_α	$U_\alpha \in \mathbb{R}^{|\mathcal{C}| \times p}$ is the block vector robot α transmits to the server in Algorithm 8.3	Alg. 8.3, line 4
ϵ	The spectral sparsification parameter that is used in the algorithm and appears in Theorem 8.2	
ρ^{TLS}	The truncated least squares (TLS) cost function for outlier-robust estimation	(8.37)
e_{ij}	Measurement error corresponding to the measurement $(i, j) \in \mathcal{E}$ in the measurement graph	
$\overline{e}$	Threshold that specifies the maximum error of inlier measurement in TLS	
μ	Control parameter of graduated non-convexity (GNC)	
w_{ij}^{GNC}	GNC weight for measurement $(i, j) \in \mathcal{E}$ in the measurement graph	(8.38)

Algorithm D.1 Spectral Sparsification by Effective Resistance Sampling

1: **for** each edge $(i, j) \in \mathcal{E}$ in the graph G corresponding to the input Laplacian L (in parallel) **do**
2: Compute leverage score $\ell_{ij} = w_{ij}(\Delta_i - \Delta_j)^\top L^\dagger(\Delta_i - \Delta_j)$.
3: Select this edge with probability $p_{ij} = \min(1,\ 3.5 \log n \ell_{ij}/\epsilon_l^2)$.
4: If this edge is selected, add it to the sparsified graph $\widetilde{G}$ with increased edge weight w_{ij}/p_{ij}.
5: **end for**
6: **return** The Laplacian $\widetilde{L}$ of the sparse graph $\widetilde{G}$.

D.1 Details of Spectral Sparsification Algorithm

In this appendix, we provide details of the spectral sparsification algorithm used in Chapter 8. Given the Laplacian matrix L of a dense graph G, recall that the goal of spectral sparsification is to find a *sparse* Laplacian $\widetilde{L}$ such that,

$$e^{-\epsilon} L \preceq \widetilde{L} \preceq e^{\epsilon} L, \tag{D.1}$$

where $\epsilon > 0$ is the desired sparsification parameter. Note that (D.1) is the same definition as (8.1). In graph terms, this is the same as finding a sparse graph $\widetilde{G}$ whose Laplacian approximates that of the dense graph G.

In this work, we use the random sampling approach developed by Spielman and Srivastava [136]. In particular, we implement the improved version presented in [217, Chapter 32], which is more suitable for batch computation since it avoids sampling with replacement and the decision to keep or remove each edge can be made in parallel. Given a constant $\epsilon_l \in (0, 1)$, this method produces a sparse $\widetilde{L}$ with $O(n \log n)$ entries (where n is the number of vertices in the graph) such that with high probability,

$$(1 - \epsilon_l) L \preceq \widetilde{L} \preceq (1 + \epsilon_l) L. \tag{D.2}$$

Note that (D.2) can be used to ensure that (D.1) holds: in our implementation, given ϵ, we find the smallest ϵ_l such that (D.1) holds, which is given by,

$$\epsilon_l = \min(e^{\epsilon} - 1, 1 - e^{-\epsilon}). \tag{D.3}$$

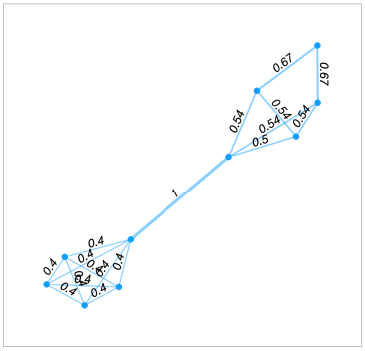


Figure D-1: Illustration of leverage scores on a toy graph.

The sparsification algorithm works by selecting edges in the input dense graph G based on their *leverage scores*. Recall that each edge corresponds to a non-zero off-diagonal term in the Laplacian L, and thus selecting a small subset of edges leads to a sparse output Laplacian $\widetilde{L}$. For each edge $(i, j) \in \mathcal{E}$, its leverage score is defined as,

$$\ell_{ij} \triangleq w_{ij}(\Delta_i - \Delta_j)^\top L^\dagger (\Delta_i - \Delta_j), \tag{D.4}$$

where $\Delta_i \in \mathbb{R}^n$ is the i-th basis vector with a one in coordinate i, and $w_{ij} > 0$ is the edge weight. The quantity ℓ_{ij}/w_{ij} is also known as the *effective resistance*. Intuitively, the leverage score measures the importance of each edge to the connectivity of the overall graph. Figure D-1 shows an illustration on a toy graph consisting of two clusters of vertices connected by a single edge. All edges have unit weights, and each edge is labeled by its leverage score computed according to (D.4). Notice that the middle edge has the highest leverage score, since it is critical to keep the overall graph connected. In comparison, the remaining edges have lower leverage scores, due to the redundancy of edges in each cluster. The actual sparsification algorithm is a remarkably simple method, which independently selects each edge with a probability proportional to its leverage score:

$$p_{ij} = \min\left(1,\ \frac{3.5 \log n}{\epsilon_l^2} \ell_{ij}\right). \tag{D.5}$$

If edge $(i, j) \in \mathcal{E}$ is selected, we add it to the sparsified graph $\widetilde{G}$ with an *increased edge*

weight w_{ij}/p_{ij}. The reason behind increasing the edge weight is to ensure that we can recover L in expectation, *i.e.*, $\mathbb{E}(\widetilde{L}) = L$. Algorithm D.1 shows the pseudocode. The majority of computation is spent on factorizing L to compute the leverage scores in (D.4). The approximation guarantee (D.2) of the resulting $\widetilde{L}$ is proved using certain concentration bounds of random matrices. The interested reader is referred to [217, Chapter 32] for the complete proof.

Lastly, we refer the reader to Figure 8-12, which demonstrates spectral sparsification on Laplacian matrices encountered in our application. Recall that in our case, we apply sparsification on the Schur complement S_α of robot α's local Laplacian. Since Laplacians are closed under Schur complements [141, Fact 4.2], S_α is still a Laplacian matrix and thus the sparsified result $\widetilde{S}_\alpha$ retains all the theoretical guarantees.

D.2 Analysis of Riemannian Hessian of Rotation Averaging

In this appendix, we use $\mathbf{d}_\angle \equiv \mathbf{d}$ to denote the geodesic distance on the rotation group, and use $\mathbf{d}_{\text{chr}}$ to denote the chordal distance. Recall the definition of φ in Problem 8.1 as either the squared geodesic distance or the squared chordal distance,

$$\varphi(R_1, R_2) = \begin{cases} \frac{1}{2}\,\mathbf{d}_\angle(R_1, R_2)^2 = \frac{1}{2}\left\|\text{Log}(R_1^\top R_2)\right\|^2, & \text{(D.6a)} \\ \frac{1}{2}\,\mathbf{d}_{\text{chr}}(R_1, R_2)^2 = \frac{1}{2}\left\|R_1 - R_2\right\|_F^2. & \text{(D.6b)} \end{cases}$$

Using the notion of "reshaped distance" introduced in [29], we may express both cases as a function of the geodesic distance as follows,

$$\varphi(R_1, R_2) = \rho(\mathbf{d}_\angle(R_1, R_2)). \tag{D.7}$$

Note that the notation for reshaped distance ρ is not to be confused with ρ^{TLS} in Section 8.4.4, which instead denotes the truncated least squares function. It can be

shown that the scalar function $\rho(\cdot)$ is defined as,

$$\rho(\theta) = \begin{cases} \theta^2/2, & \text{for squared geodesic distance (D.6a),} \quad \text{(D.8a)} \\ 2 - 2\cos(\theta), & \text{for squared chordal distance (D.6b).} \quad \text{(D.8b)} \end{cases}$$

The first case (D.8a) is readily verified. The second case (D.8b) makes use of the relation between chordal and geodesic distances; see [30, Table 2]. To analyze the Hessian of rotation averaging (8.5), we start by considering the cost associated with a single relative rotation measurement,

$$f_{ij}(R_i, R_j) = \varphi(R_i \widetilde{R}_{ij}, R_j). \tag{D.9}$$

In addition to (D.9), we also consider its approximation defined on the Lie algebra,

$$h_{ij}(v_i, v_j) \triangleq \varphi(\mathrm{Exp}(v_i) R_i \widetilde{R}_{ij}, \mathrm{Exp}(v_j) R_j). \tag{D.10}$$

Note that (D.10) corresponds to a single term in the overall approximation defined in (8.10). Similar to (8.10), h_{ij} depends on the current rotation estimates R, but we omit this from our notation for simplicity. Define the gradient and Hessian of h_{ij} as follows,

$$\overline{g}_{ij} \triangleq \nabla h_{ij}(v_i, v_j)|_{v_i = v_j = 0}, \tag{D.11}$$

$$\overline{H}_{ij} \triangleq \nabla^2 h_{ij}(v_i, v_j)|_{v_i = v_j = 0}. \tag{D.12}$$

We prove the main theoretical results Theorem 8.1 and Corollary 8.1 for 3D rotation averaging. The case of $d = 2$ can be proved using the exact same arguments, and some steps would simplify due to the fact that 2D rotations commute. In the following, we first derive auxiliary results that characterize $\overline{g}_{ij}$ and $\overline{H}_{ij}$. Once we understand the properties of $\overline{g}_{ij}$ and $\overline{H}_{ij}$, understanding the full rotation averaging problem becomes straightforward thanks to the additive structure in the cost function (8.5).

D.2.1 Auxiliary Results for 3D Rotation Averaging

Lemma D.1. *Consider a 3D rotation averaging problem. Let*

$$\theta_{ij} = \left\| \text{Log}(\widetilde{R}_{ij}^\top R_i^\top R_j) \right\|, \tag{D.13}$$

$$u_{ij} = \text{Log}(\widetilde{R}_{ij}^\top R_i^\top R_j)/\theta_{ij}, \tag{D.14}$$

denote the angle-axis representation of the current rotation error. Then the gradient is given by,

$$\overline{g}_{ij} = \dot{\rho}(\theta_{ij}) \begin{bmatrix} R_i \widetilde{R}_{ij} & 0 \\ 0 & R_j \end{bmatrix} \begin{bmatrix} -u_{ij} \\ u_{ij} \end{bmatrix}. \tag{D.15}$$

The Hessian is given by,

$$\overline{H}_{ij} = \begin{bmatrix} R_i \widetilde{R}_{ij} & 0 \\ 0 & R_j \end{bmatrix} \begin{bmatrix} \mathcal{S}(\widetilde{H}_{ij}) & -\widetilde{H}_{ij} \\ -\widetilde{H}_{ij}^\top & \mathcal{S}(\widetilde{H}_{ij}) \end{bmatrix} \begin{bmatrix} R_i \widetilde{R}_{ij} & 0 \\ 0 & R_j \end{bmatrix}^\top, \tag{D.16}$$

where $\widetilde{H}_{ij} = \alpha(\theta_{ij}) I + \gamma(\theta_{ij}) u_{ij} u_{ij}^\top + \beta(\theta_{ij}) [u_{ij}]_\times$ *with*

$$\alpha(\theta_{ij}) = \frac{\dot{\rho}(\theta_{ij}) \cot(\theta_{ij}/2)}{2}, \tag{D.17}$$

$$\gamma(\theta_{ij}) = \ddot{\rho}(\theta_{ij}) - \alpha(\theta_{ij}), \tag{D.18}$$

$$\beta(\theta_{ij}) = \frac{\dot{\rho}(\theta_{ij})}{2}. \tag{D.19}$$

Proof. Introduce new rotation variables $S_i, S_j \in \text{SO}(3)$ and consider the following function,

$$\widehat{h}_{ij}(S_i, S_j) = \varphi(S_i R_i \widetilde{R}_{ij}, S_j R_j). \tag{D.20}$$

Note that $\overline{g}_{ij}$ and $\overline{H}_{ij}$ correspond to the Riemannian gradient and Riemannian Hessian of (D.20) evaluated at $S_i = S_j = I$. Define $F : \text{SO}(3) \times \text{SO}(3) \to \text{SO}(3) \times \text{SO}(3)$ be

the mapping such that,

$$F(S_i, S_j) = (S_i R_i \widetilde{R}_{ij}, S_j R_j) \triangleq (\widehat{S}_i, \widehat{S}_j). \tag{D.21}$$

Then we have $\widehat{h}_{ij}(S_i, S_j) = \varphi(F(S_i, S_j))$. By chain rule,

$$\begin{aligned} \operatorname{grad} \widehat{h}_{ij}(S_i, S_j) &= DF(S_i, S_j)^\top [\operatorname{grad} \varphi(\widehat{S}_i, \widehat{S}_j)] && \text{(D.22)} \\ &= DF(S_i, S_j)^\top [\dot{\rho}(\theta_{ij}) \operatorname{grad} \mathbf{d}_\angle(\widehat{S}_i, \widehat{S}_j)] && \text{(D.23)} \\ &= \dot{\rho}(\theta_{ij}) DF(S_i, S_j)^\top [\operatorname{grad} \mathbf{d}_\angle(\widehat{S}_i, \widehat{S}_j)]. && \text{(D.24)} \end{aligned}$$

In (D.24), $DF(S_i, S_j)^\top$ stands for the adjoint operator (transpose in matrix form) of the differential $DF(S_i, S_j)$. Using the standard basis for the Lie algebra so(3), Tron [29] showed that the Riemannian gradient of the geodesic distance is,

$$\operatorname{grad} \mathbf{d}_\angle(\widehat{S}_i, \widehat{S}_j) = \begin{bmatrix} -u_{ij} \\ u_{ij} \end{bmatrix}, \tag{D.25}$$

c.f. [29, Equation (E.31)]. Substituting (D.25) into (D.24) and furthermore using the matrix form of the differential $DF(S_i, S_j)$ in the standard basis, we have

$$\overline{g}_{ij} = \operatorname{grad} \widehat{h}_{ij}(S_i, S_j) = \dot{\rho}(\theta_{ij}) \begin{bmatrix} R_i \widetilde{R}_{ij} & 0 \\ 0 & R_j \end{bmatrix} \begin{bmatrix} -u_{ij} \\ u_{ij} \end{bmatrix}. \tag{D.26}$$

For the Riemannian Hessian, differentiating (D.22) again yields,

$$\operatorname{Hess} \widehat{h}_{ij}(S_i, S_j) = DF(S_i, S_j)^\top \circ \operatorname{Hess} \varphi(\widehat{S}_i, \widehat{S}_j) \circ DF(S_i, S_j). \tag{D.27}$$

For the Hessian term in the middle of (D.27), we once again leverage existing results from [29, Proposition E.3.1]:

$$\operatorname{Hess} \varphi(\widehat{S}_i, \widehat{S}_j) = \begin{bmatrix} \mathcal{S}(\widetilde{H}_{ij}) & -\widetilde{H}_{ij} \\ -\widetilde{H}_{ij}^\top & \mathcal{S}(\widetilde{H}_{ij}) \end{bmatrix}, \tag{D.28}$$

where the inner matrix $\widetilde{H}_{ij}$ is defined as,

$$\widetilde{H}_{ij} = \ddot{\rho}(\theta_{ij}) u_{ij} u_{ij}^\top + \frac{\dot{\rho}(\theta_{ij})}{\theta_{ij}} (D \operatorname{Log}(\widetilde{R}_{ij}^\top R_i^\top R_j) - u_{ij} u_{ij}^\top). \tag{D.29}$$

Using the expression for the differential of the logarithm map [29, Proposition E.2.1], the previous expression further simplifies to,

$$\begin{aligned}
\widetilde{H}_{ij} &= \ddot{\rho}(\theta_{ij}) u_{ij} u_{ij}^\top + \frac{\dot{\rho}(\theta_{ij})}{\theta_{ij}} \left(u_{ij} u_{ij}^\top + \frac{\theta_{ij}}{2} \left([u_{ij}]_\times - \cot(\theta_{ij}/2) [u_{ij}]_\times^2 \right) - u_{ij} u_{ij}^\top \right) \\
&= \ddot{\rho}(\theta_{ij}) u_{ij} u_{ij}^\top + \frac{\dot{\rho}(\theta_{ij})}{2} \left([u_{ij}]_\times - \cot(\theta_{ij}/2) [u_{ij}]_\times^2 \right) \\
&= \ddot{\rho}(\theta_{ij}) u_{ij} u_{ij}^\top + \frac{\dot{\rho}(\theta_{ij})}{2} [u_{ij}]_\times - \frac{\dot{\rho}(\theta_{ij}) \cot(\theta_{ij}/2)}{2} \left(-I + u_{ij} u_{ij}^\top \right) \\
&= \frac{\dot{\rho}(\theta_{ij}) \cot(\theta_{ij}/2)}{2} I + \left(\ddot{\rho}(\theta_{ij}) - \frac{\dot{\rho}(\theta_{ij}) \cot(\theta_{ij}/2)}{2} \right) u_{ij} u_{ij}^\top + \frac{\dot{\rho}(\theta_{ij})}{2} [u_{ij}]_\times \\
&= \alpha(\theta_{ij}) I + \gamma(\theta_{ij}) u_{ij} u_{ij}^\top + \beta(\theta_{ij}) [u_{ij}]_\times .
\end{aligned} \tag{D.30}$$

To conclude, the Hessian is obtained by substituting the above results into (D.27):

$$\overline{H}_{ij} = \operatorname{Hess} \widehat{h}_{ij}(S_i, S_j) = \begin{bmatrix} R_i \widetilde{R}_{ij} & 0 \\ 0 & R_j \end{bmatrix} \begin{bmatrix} \mathcal{S}(\widetilde{H}_{ij}) & -\widetilde{H}_{ij} \\ -\widetilde{H}_{ij}^\top & \mathcal{S}(\widetilde{H}_{ij}) \end{bmatrix} \begin{bmatrix} R_i \widetilde{R}_{ij} & 0 \\ 0 & R_j \end{bmatrix}^\top . \tag{D.31}$$

□

The Hessian expression in Lemma D.1 is complicated in general. However, we will show that as the angular error θ_{ij} tends to zero, the Hessian $\overline{H}_{ij}$ converges to a particular simple form. We note that the case under geodesic distance (Lemma D.2 below) can also be derived as a special case of [126, Theorem 1].

Lemma D.2 (Limit of $\overline{H}_{ij}$ under geodesic distance)**.** *For rotation averaging under the geodesic distance, it holds that,*

$$\lim_{\theta_{ij} \to 0} \overline{H}_{ij}(\theta_{ij}) = \begin{bmatrix} I_3 & -I_3 \\ -I_3 & I_3 \end{bmatrix} . \tag{D.32}$$

Proof. We first compute limits of $\alpha(\theta), \gamma(\theta)$, and $\beta(\theta)$ that appear in the definition of $\overline{H}_{ij}$. For rotation averaging under the geodesic distance, the scalar function $\rho(\theta)$ is defined as in (D.8a). In this case, we have

$$\dot{\rho}(\theta) = \theta, \quad \ddot{\rho}(\theta) = 1. \tag{D.33}$$

Substituting into (D.17)-(D.19),

$$\alpha(\theta) = \frac{1}{2}\theta \cot(\theta/2) = \frac{1}{2}\frac{\theta}{\sin(\theta/2)}\cos(\theta/2), \tag{D.34}$$

$$\gamma(\theta) = 1 - \alpha(\theta), \tag{D.35}$$

$$\beta(\theta) = \theta/2. \tag{D.36}$$

Take the limit as θ tends to zero,

$$\lim_{\theta\to 0}\alpha(\theta) = \frac{1}{2}\cdot\lim_{\theta\to 0}\frac{\theta}{\sin(\theta/2)}\cdot\lim_{\theta\to 0}\cos(\theta/2) = 1, \tag{D.37}$$

$$\lim_{\theta\to 0}\gamma(\theta) = 1 - \lim_{\theta\to 0}\alpha(\theta) = 0, \tag{D.38}$$

$$\lim_{\theta\to 0}\beta(\theta) = 0. \tag{D.39}$$

Define the following matrix,

$$P = \begin{bmatrix} R_i\widetilde{R}_{ij} & 0 \\ 0 & R_j \end{bmatrix}. \tag{D.40}$$

From the definition of $\overline{H}_{ij}$ in (D.16),

$$\begin{aligned} \overline{H}_{ij} = {} & \alpha(\theta_{ij})P\begin{bmatrix} I_3 & -I_3 \\ -I_3 & I_3 \end{bmatrix}P^\top \\ & + \gamma(\theta_{ij})P\begin{bmatrix} u_{ij}u_{ij}^\top & -u_{ij}u_{ij}^\top \\ -u_{ij}u_{ij}^\top & u_{ij}u_{ij}^\top \end{bmatrix}P^\top \\ & + \beta(\theta_{ij})P\begin{bmatrix} 0_3 & -[u_{ij}]_\times \\ -[u_{ij}]_\times^\top & 0_3 \end{bmatrix}P^\top. \end{aligned} \tag{D.41}$$

Since $\lim_{\theta\to 0}\gamma(\theta) = \lim_{\theta\to 0}\beta(\theta) = 0$ and all matrices involved in (D.41) are bounded, we conclude that the last two terms in (D.41) vanish as θ_{ij} converges to zero. For the first term in (D.41), notice that,

$$\alpha(\theta_{ij})P\begin{bmatrix} I_3 & -I_3 \\ -I_3 & I_3 \end{bmatrix}P^\top = \alpha(\theta_{ij})\begin{bmatrix} I_3 & -R_i\widetilde{R}_{ij}R_j^\top \\ -R_j\widetilde{R}_{ij}^\top R_i^\top & I_3 \end{bmatrix}. \tag{D.42}$$

As θ_{ij} tends to zero, $\alpha(\theta_{ij})$ converges to 1 and the off-diagonal blocks in (D.42) tend to $-I_3$. Hence the proof is completed. □

Lemma D.3 (Limit of $\overline{H}_{ij}$ under chordal distance)**.** *For rotation averaging under the chordal distance, it holds that,*

$$\lim_{\theta_{ij}\to 0}\overline{H}_{ij}(\theta_{ij}) = 2\begin{bmatrix} I_3 & -I_3 \\ -I_3 & I_3 \end{bmatrix}. \tag{D.43}$$

Proof. For rotation averaging under the chordal distance, the scalar function $\rho(\theta)$ is defined as in (D.8b). In this case, we have

$$\dot{\rho}(\theta) = 2\sin(\theta), \quad \ddot{\rho}(\theta) = 2\cos(\theta). \tag{D.44}$$

Substituting into (D.17)-(D.19),

$$\alpha(\theta) = \sin(\theta)\cot(\theta/2) = 2\cos(\theta/2)^2, \tag{D.45}$$

$$\gamma(\theta) = 2\cos(\theta) - \alpha(\theta), \tag{D.46}$$

$$\beta(\theta) = \sin(\theta). \tag{D.47}$$

Take the limit as θ tends to zero,

$$\lim_{\theta \to 0} \alpha(\theta) = 2, \tag{D.48}$$

$$\lim_{\theta \to 0} \gamma(\theta) = 2 - \lim_{\theta \to 0} \alpha(\theta) = 0, \tag{D.49}$$

$$\lim_{\theta \to 0} \beta(\theta) = 0. \tag{D.50}$$

The remaining proof is similar to that of Lemma D.2 and is omitted. □

D.2.2 Proof of Theorem 8.1

Proof. We will use Lemma D.2 to prove the theorem for the case of squared geodesic cost. The case of squared chordal cost is analogous: instead of Lemma D.2, we will use Lemma D.3 and the remaining steps are the same. Recall the approximation of the overall cost function defined in (8.10):

$$h(v; R) = \sum_{(i,j) \in \mathcal{E}} \kappa_{ij} h_{ij}(v_i, v_j) = \sum_{(i,j) \in \mathcal{E}} \kappa_{ij} \varphi(\mathrm{Exp}(v_i) R_i \widetilde{R}_{ij}, \mathrm{Exp}(v_j) R_j), \tag{D.51}$$

Observe that the Hessian of $h(v; R)$ is simply given by the sum of the Hessian matrices of $h_{ij}(v_i, v_j)$, after "lifting" the latter to the dimension of the full optimization problem, *i.e.*,

$$\overline{H}(R) = \sum_{(i,j) \in \mathcal{E}} \kappa_{ij} W_{ij}, \quad W_{ij} = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} & & i & & j \end{array} \\ \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & \overline{H}_{ij}^{(ii)} & \cdots & \overline{H}_{ij}^{(ij)} & \cdots \\ & \vdots & & \vdots & \\ \cdots & \overline{H}_{ij}^{(ji)} & \cdots & \overline{H}_{ij}^{(jj)} & \cdots \\ & \vdots & & \vdots & \end{bmatrix} \end{array}. \tag{D.52}$$

In (D.52), W_{ij} is formed by placing the p-by-p blocks of $\overline{H}_{ij}$ defined in (D.16) in corresponding locations of the full matrix. For instance, $\overline{H}_{ij}^{(ii)}$ is the block of $\overline{H}_{ij}$ that corresponds to vertex i.

In the following, let $\theta_{ij}(R)$ denote the residual of edge $(i,j) \in \mathcal{E}$ evaluated at $R \in \mathrm{SO}(d)^n$. From (D.52) and Lemma D.2, we see that $\overline{H}(R)$ has the following limit point as all edge residuals tend to zero,

$$\lim_{\substack{\theta_{ij}(R)\to 0,\\ \forall (i,j)\in\mathcal{E}}} \overline{H}(R) = M \triangleq L(G;\kappa) \otimes I_p. \tag{D.53}$$

Recall from Remark 8.2 that $M \succeq 0$ and $\ker(M) = \mathcal{N}$ where $\mathcal{N}$ is the vertical space defined in (8.11). In addition, recall the definition of $H(R)$ in (8.13):

$$H(R) = P_H \overline{H}(R) P_H. \tag{D.54}$$

Since P_H is the (constant) orthogonal projection matrix onto the horizontal space $\mathcal{H} = \mathcal{N}^\perp$, it holds that,

$$\lim_{\substack{\theta_{ij}(R)\to 0,\\ \forall (i,j)\in\mathcal{E}}} H(R) = P_H \, M \, P_H = M. \tag{D.55}$$

Note that for singular symmetric matrices A and B, $A \approx_\delta B$ necessarily means that $\ker(A) = \ker(B)$. Therefore, to prove the theorem, we must first show that $\ker(H(R)) = \ker(M) = \mathcal{N}$ under our assumptions. Let $\lambda_1(A), \lambda_2(A), \ldots$ denote the eigenvalues of a symmetric matrix A sorted in increasing order. By construction, $\mathcal{N}$ is always contained in $\ker(H(R))$, and thus $\dim(\mathcal{N}) = p$ eigenvalues of $H(R)$ are always zero. Next, we will show that if all measurement residuals are sufficiently small, then the remaining $pn - p$ eigenvalues of $H(R)$ will be strictly positive. Define $E(R) \triangleq H(R) - M$. Let x be any unit vector such that $x \perp \mathcal{N}$. Note that,

$$\begin{aligned} x^\top H(R) x &= x^\top M x + x^\top E(R) x \\ &\geq \lambda_{p+1}(M) - \|E(R)\|_2 \,. \end{aligned} \tag{D.56}$$

Since $M = L(G;\kappa) \otimes I_p$, it holds that $\lambda_{p+1}(M) = \lambda_2(L(G;\kappa))$. The latter is known as the *algebraic connectivity* which is always positive for a connected graph G. Thus $\lambda_{p+1}(M) > 0$ and by (D.55), we also have $\lim_{\theta_{ij}(R)\to 0} E(R) = 0$. Consequently, when

all $\theta_{ij}(R)$ are sufficiently small, the right-hand side of (D.56) is strictly positive, *i.e.*, there exists $\bar{\theta}_1 > 0$ such that if $\theta_{ij}(R) \leq \bar{\theta}_1$ for all $(i,j) \in \mathcal{E}$, we have,

$$\ker(H(R)) = \ker(M) = \mathcal{N}. \tag{D.57}$$

Under (D.57), the desired approximation $H(R) \approx_\delta M$ is equivalent to,

$$e^{-\delta} P_H \preceq M^{\frac{+}{2}} H(R) M^{\frac{+}{2}} \preceq e^{\delta} P_H, \tag{D.58}$$

where $M^{\frac{+}{2}}$ denotes the square root of the pseudoinverse of M, and P_H is the orthogonal projection onto the horizontal space $\mathcal{H}$. This condition is true if and only if the nontrivial eigenvalues are bounded as follows,

$$\lambda_{p+1}(M^{\frac{+}{2}} H(R) M^{\frac{+}{2}}) \geq e^{-\delta},\ \lambda_{pn}(M^{\frac{+}{2}} H(R) M^{\frac{+}{2}}) \leq e^{\delta}. \tag{D.59}$$

Using the convergence result (D.55) and the eigenvalue perturbation bounds in [218, Corollary 6.3.8], we conclude that there exists $\bar{\theta}_0 \in (0, \bar{\theta}_1]$ such that if $R \in \mathrm{SO}(d)^n$ satisfies

$$\theta_{ij}(R) \leq \bar{\theta}_0,\ \forall (i,j) \in \mathcal{E}, \tag{D.60}$$

then (D.59) holds, *i.e.*, we have the desired approximation,

$$H(R) \approx_\delta M. \tag{D.61}$$

To conclude the proof, we need to show that there exist $\bar{\theta}, r > 0$ such that condition (D.60) holds for all $R \in B_r(R^\star)$. Let us first consider residuals at the global minimizer $R^\star$. Using assumption (8.14) and the cost function, we obtain the following simple bound:

$$\max_{(i,j)\in\mathcal{E}} \frac{\kappa_{ij} \theta_{ij}(R^\star)^2}{2} \leq f(R^\star) \leq f(\underline{R}) \leq \frac{\sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \bar{\theta}^2}{2}. \tag{D.62}$$

It can be verified that if,

$$\bar{\theta} \leq \frac{\bar{\theta}_0}{2} \sqrt{\frac{\min_{(i,j)\in\mathcal{E}} \kappa_{ij}}{\sum_{(i,j)\in\mathcal{E}} \kappa_{ij}}}, \tag{D.63}$$

then (D.62) yields $\theta_{ij}(R^\star) \leq \bar{\theta}_0/2$ for all edges $(i,j) \in \mathcal{E}$. Finally, let us select $r \in (0,\ \bar{\theta}_0/4)$. For $i \in [n]$, let $E_i \in \mathrm{SO}(d)$ such that $R_i = E_i R_i^\star$. Using the triangle inequality and the fact that the geodesic distance $\mathbf{d}_\angle(\cdot,\cdot)$ is bi-invariant, we can show that for any $R \in B_r(R^\star)$,

$$\begin{aligned}
\theta_{ij}(R) &= \mathbf{d}_\angle(R_i \widetilde{R}_{ij}, R_j) && \text{(D.64)}\\
&= \mathbf{d}_\angle(E_i R_i^\star \widetilde{R}_{ij}, E_j R_j^\star) && \text{(D.65)}\\
&= \mathbf{d}_\angle(R_i^\star \widetilde{R}_{ij} (R_j^\star)^\top, E_i^\top E_j) && \text{(D.66)}\\
&\leq \mathbf{d}_\angle(R_i^\star \widetilde{R}_{ij} (R_j^\star)^\top, I) + \mathbf{d}_\angle(E_i, I) + \mathbf{d}_\angle(E_j, I) && \text{(D.67)}\\
&\leq \theta_{ij}(R^\star) + 2r && \text{(D.68)}\\
&\leq \bar{\theta}_0. && \text{(D.69)}
\end{aligned}$$

In summary, we have shown that if $\bar{\theta}$ satisfies (D.63) and furthermore $0 < r < \bar{\theta}_0/4$, then the desired approximation $M \approx_\delta H(R)$ holds for all $R \in B_r(R^\star)$. This concludes the proof. □

D.2.3 Proof of Corollary 8.1

Proof. To simplify notation, we use L to denote $L(G; w)$. By (8.15), it holds that,

$$e^{-\delta}(L \otimes I_p) \preceq H(R) \preceq e^{\delta}(L \otimes I_p). \tag{D.70}$$

Note that the eigenvalues of $L \otimes I_p$ are given by the eigenvalues of L, repeated p times. Therefore, the desired result follows by noting that,

$$\lambda_2(L) P_H \preceq L \otimes I_p \preceq \lambda_n(L) P_H. \tag{D.71}$$

□

D.3 Performance Guarantees for Collaborative Laplacian Solver

D.3.1 Proof of Lemma 8.1

Proof. By definition in (8.25),

$$S = L_{cc} - \sum_{\alpha \in [m]} L_{c\alpha} L_{\alpha\alpha}^{-1} L_{\alpha c}. \tag{D.72}$$

Above, L_{cc} is the block of the full Laplacian L that corresponds to the separators, denoted as $L_{cc} \equiv L(G)_{cc}$. Note that L_{cc} can be decomposed as the sum,

$$L_{cc} = L(G_c) + \sum_{\alpha \in [m]} L(G_\alpha)_{cc}. \tag{D.73}$$

Intuitively, the first term in (D.73) accounts for inter-robot edges $\mathcal{E}_c$, and the second group of terms accounts for robots' local edges $\mathcal{E}_\alpha$; see Figure 8-1a. Substitute (D.73) into (D.72),

$$\begin{aligned} S &= L(G_c) + \sum_{\alpha \in [m]} \left(L(G_\alpha)_{cc} - L_{c\alpha} L_{\alpha\alpha}^{-1} L_{\alpha c} \right) \\ &= L(G_c) + \sum_{\alpha \in [m]} \mathrm{Sc}(L(G_\alpha), \mathcal{F}_\alpha). \end{aligned} \tag{D.74}$$

□

D.3.2 Proof of Theorem 8.2

Proof. Let us simplify the notations in the Laplacian system (8.24) by considering interior nodes from all robots as a single block:

$$\begin{bmatrix} L_{ff} & L_{fc} \\ L_{cf} & L_{cc} \end{bmatrix} \begin{bmatrix} X_f \\ X_c \end{bmatrix} = \begin{bmatrix} B_f \\ B_c \end{bmatrix} \tag{D.75}$$

where

$$L_{ff} = \mathrm{Diag}(L_{11}, \ldots, L_{mm}), \tag{D.76}$$

$$L_{cf} = L_{fc}^{\top} = \begin{bmatrix} L_{c1} & \ldots & L_{cm} \end{bmatrix}, \tag{D.77}$$

$$X_f = \begin{bmatrix} X_1^{\top} & \ldots & X_m^{\top} \end{bmatrix}^{\top}, \tag{D.78}$$

$$B_f = \begin{bmatrix} B_1^{\top} & \ldots & B_m^{\top} \end{bmatrix}^{\top}. \tag{D.79}$$

By applying the Schur complement to (D.75), we obtain the following factorization for the input Laplacian system $LX = B$,

$$\underbrace{\begin{bmatrix} I & 0 \\ L_{cf}L_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} L_{ff} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & L_{ff}^{-1}L_{fc} \\ 0 & I \end{bmatrix}}_{L} \begin{bmatrix} X_f \\ X_c \end{bmatrix} = \begin{bmatrix} B_f \\ B_c \end{bmatrix}, \tag{D.80}$$

where $S = \mathrm{Sc}(L, \mathcal{F})$ is the Schur complement that appears in (8.25). It can be verified that Algorithm 8.3 returns a solution to the following system,

$$\underbrace{\begin{bmatrix} I & 0 \\ L_{cf}L_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} L_{ff} & 0 \\ 0 & \widetilde{S} \end{bmatrix} \begin{bmatrix} I & L_{ff}^{-1}L_{fc} \\ 0 & I \end{bmatrix}}_{\widetilde{L}} \begin{bmatrix} X_f \\ X_c \end{bmatrix} = \begin{bmatrix} B_f \\ B_c \end{bmatrix}. \tag{D.81}$$

Recall from Lemma 8.1 that,

$$S = L(G_c) + \sum_{\alpha \in [m]} S_\alpha. \tag{D.82}$$

Meanwhile, by construction, $\widetilde{S}$ is given by,

$$\widetilde{S} = L(G_c) + \sum_{\alpha \in [m]} \widetilde{S}_\alpha, \tag{D.83}$$

where $\widetilde{S}_\alpha \approx_\epsilon S_\alpha$ for all $\alpha \in [m]$. Since spectral approximation is preserved under addition, it holds that $\widetilde{S} \approx_\epsilon S$. Furthermore, by comparing L defined in (D.80) and $\widetilde{L}$

Algorithm D.2 APPROXIMATE NEWTON METHOD

1: **for** iteration $k = 0, 1, \ldots$ **do**
2: $\quad \eta^k = -M(x^k)^{-1} \operatorname{grad} f(x^k)$.
3: $\quad$ Update iterate by $x^{k+1} = \operatorname{Retr}_{x^k}(\eta^k)$.
4: **end for**

defined in (D.81) and using [141, Fact 3.2], we conclude that $\widetilde{L} \approx_\epsilon L$ and thus (8.29) is true. Lastly, (8.30) follows from Lemma D.4. □

D.4 Convergence Analysis

In this section, we establish convergence guarantees for the collaborative rotation averaging (Algorithm 8.4) and translation estimation (Algorithm 8.5) methods developed in Section 8.4. Between the two, analyzing Algorithm 8.4 is more complicated owing to the fact that rotation averaging is an optimization problem defined on a Riemannian manifold. To establish its convergence, in appendix D.4.1 we first prove a more general result that holds for generic approximate Newton methods on manifolds. Then, in appendix D.4.2, we invoke this result for the special case of rotation averaging and show that Algorithm 8.4 enjoys a local *linear* convergence rate. Lastly, in appendix D.4.3, we prove the linear convergence of translation estimation (Algorithm 8.5).

D.4.1 Analysis of General Approximate Newton Method

In this subsection, we consider a generic optimization problem on a smooth Riemannian manifold $\mathcal{M}$:

$$\min_{x \in \mathcal{M}} f(x). \tag{D.84}$$

We consider solving the above problem using an *approximate Newton method* described in Algorithm D.2. At each iteration, the Riemannian Hessian $\operatorname{Hess} f(x)$ is replaced with an approximation $M(x)$, and the update is computed by solving a linear system in $M(x)$; see line 2. We will show that under the following assumptions (in particular, $M(x)$ is a sufficiently good approximation of $\operatorname{Hess} f(x)$), Algorithm D.2

achieves a local linear rate of convergence.

Assumption D.1. Let $x^\star$ denote a strict second-order critical point. There exist $\mu_H, L_H, \beta, \epsilon > 0$ such that for all x in a neighborhood $\mathcal{U}$ of $x^\star$,

(A1) $\mu_H I \preceq \operatorname{Hess} f(x) \preceq L_H I$.

(A2) $M(x)$ is invertible and $\|M(x)^{-1}\| \leq \beta$.

(A3) $M(x) \approx_\epsilon \operatorname{Hess} f(x)$ and ϵ satisfies

$$\gamma(\epsilon) \triangleq 2\sqrt{\kappa_H} c(\epsilon) < 1, \tag{D.85}$$

where $c(\epsilon)$ is defined in (8.31) and $\kappa_H = L_H/\mu_H$ is the condition number.

Theorem D.1. *Under Assumption D.1, there exists a neighborhood $\mathcal{U}' \subseteq \mathcal{U}$ such that for all $x_0 \in \mathcal{U}'$, Algorithm D.2 generates an infinite sequence x^k converging linearly to $x^\star$. Furthermore, the linear convergence factor is given by,*

$$\limsup_{k\to\infty} \frac{\mathbf{d}(x^{k+1}, x^\star)}{\mathbf{d}(x^k, x^\star)} = \gamma(\epsilon). \tag{D.86}$$

Proof. We prove the theorem by adapting the local convergence analysis of the Riemannian Newton method presented in [17, Theorem 6.3.2]. Let $(\mathcal{U}', \varphi)$ be a local coordinate chart defined by the normal coordinates around $x^\star$. Similar to the original proof, we will use the $\hat{\cdot}$ notation to denote coordinate expressions in this chart. In particular, let us define,

$$\widehat{x} = \varphi(x) = \operatorname{Exp}_{x^\star}^{-1}(x). \tag{D.87}$$

Note that under the normal coordinates, we have $\widehat{x^\star} = 0$, $\operatorname{Exp}_{x^\star}(\widehat{x}) = x$, and

$\mathbf{d}(x, x^\star) = \|\widehat{x}\|$. In addition, let us define,

$$\widehat{\eta} = \mathrm{D}\,\varphi(x)[\eta],\ \eta \in T_x\mathcal{M}, \tag{D.88}$$

$$\widehat{g}(\widehat{x}^k) = \mathrm{D}\,\varphi(x^k)[\operatorname{grad} f], \tag{D.89}$$

$$\widehat{H}(\widehat{x}^k) = \mathrm{D}\,\varphi(x^k) \circ \operatorname{Hess} f(x^k) \circ (\mathrm{D}\,\varphi(x^k))^{-1}, \tag{D.90}$$

$$\widehat{R}_{\widehat{x}}(\widehat{\eta}) = \varphi(\operatorname{Retr}_x(\eta)),\ \eta \in T_x\mathcal{M}, \tag{D.91}$$

to be the coordinate expressions of vector fields, gradient, Hessian, and the retraction, respectively. Finally, let $\widehat{M}$ denote the coordinate expression of the linear map M used in Algorithm D.2:

$$\widehat{M}(\widehat{x}^k) = \mathrm{D}\,\varphi(x_k) \circ M(x^k) \circ (\mathrm{D}\,\varphi(x_k))^{-1}. \tag{D.92}$$

Let us express each iteration of Algorithm D.2 in the chart,

$$\widehat{x}^{k+1} = \widehat{R}_{\widehat{x}^k}(-\widehat{M}(\widehat{x}^k)^{-1}\widehat{g}(\widehat{x}^k)). \tag{D.93}$$

Using the triangle inequality, we can bound the distance between x^{k+1} and $x^\star$,

$$\begin{aligned}
\mathbf{d}(x^{k+1}, x^\star) &= \left\|\widehat{x}^{k+1} - \widehat{x}^\star\right\| \\
&= \left\|\widehat{R}_{\widehat{x}^k}(-\widehat{M}(\widehat{x}^k)^{-1}\widehat{g}(\widehat{x}^k)) - \widehat{x}^\star\right\| \\
&\leq \underbrace{\left\|\widehat{R}_{\widehat{x}^k}(-\widehat{M}(\widehat{x}^k)^{-1}\widehat{g}(\widehat{x}^k)) - (\widehat{x}^k - \widehat{M}(\widehat{x}^k)^{-1}\widehat{g}(\widehat{x}^k))\right\|}_{A} + \\
&\quad \underbrace{\left\|\widehat{x}^k - \widehat{M}(\widehat{x}^k)^{-1}\widehat{g}(\widehat{x}^k) - \widehat{x}^\star\right\|}_{B}
\end{aligned} \tag{D.94}$$

In the following, we will derive upper bounds for A and B as a function of $\left\|\widehat{x}^k - \widehat{x}^\star\right\| = \mathbf{d}(x^k, x^\star)$.

Bounding A: Note that at $x^\star$, we have $\mathrm{D}\,\varphi(x^\star) = I$. Since φ is smooth, there exists $r_1 > 0$ such that for all $x \in B_{r_1}(x^\star) = \{x \in \mathcal{M},\ \mathbf{d}(x, x^\star) < r_1\}$,

$$\|\mathrm{D}\,\varphi(x)\| \leq \sqrt{2},\ \left\|(\mathrm{D}\,\varphi(x))^{-1}\right\| \leq \sqrt{2}. \tag{D.95}$$

It follows from (A2) and (D.92) that,

$$\left\|\widehat{M}(\widehat{x})^{-1}\right\| \leq 2\beta,\ \forall x \in B_{r_1}(x^\star). \tag{D.96}$$

Furthermore, Assumption (A1) implies that the gradient is Lipschitz continuous. Using the fact that $\widehat{g}(\widehat{x}^\star) = 0$ (since $x^\star$ is a critical point), we have the following upper bound for the norm of the Newton step,

$$\begin{aligned}\left\|\widehat{M}(\widehat{x}^k)^{-1}\widehat{g}(\widehat{x}^k)\right\| &= \left\|\widehat{M}(\widehat{x}^k)^{-1}(\widehat{g}(\widehat{x}^k) - \widehat{g}(\widehat{x}^\star))\right\| \\ &\leq 2\beta L' \left\|\widehat{x}^k - \widehat{x}^\star\right\|,\end{aligned} \tag{D.97}$$

where $L' > 0$ is a fixed constant. Using the local rigidity property of the retraction (*e.g.*, see [17, Definition 4.1.1]), we have that,

$$\left\|\widehat{R}_{\widehat{x}}(\widehat{\eta}) - (\widehat{x} + \widehat{\eta})\right\| = O(\|\widehat{\eta}\|^2), \tag{D.98}$$

for all x in a neighborhood of $x^\star$ and all η sufficiently small; see also the discussions in [17, p. 115]. It follows from (D.98) and (D.97) that there exists $r_2, C_2 > 0$ such that

$$A \leq C_2 \left\|\widehat{x}^k - \widehat{x}^\star\right\|^2, \tag{D.99}$$

for all $x^k \in B_{r_2}(x^\star)$.

Bounding B: To begin, we derive the following upper bound for B using triangle inequality,

$$\begin{aligned} B &= \left\| \widehat{x}^k - \widehat{M}(\widehat{x}^k)^{-1} \widehat{g}(\widehat{x}^k) - \widehat{x}^\star \right\| \\ &= \left\| \widehat{M}(\widehat{x}^k)^{-1} \left[\widehat{g}(\widehat{x}^\star) - \widehat{g}(\widehat{x}^k) - \widehat{M}(\widehat{x}^k)(\widehat{x}^\star - \widehat{x}^k) \right] \right\| \\ &= \left\| \widehat{M}(\widehat{x}^k)^{-1} \left[\widehat{g}(\widehat{x}^\star) - \widehat{g}(\widehat{x}^k) - \widehat{H}(\widehat{x}^k)(\widehat{x}^\star - \widehat{x}^k) - (\widehat{M}(\widehat{x}^k) - \widehat{H}(\widehat{x}^k))(\widehat{x}^\star - \widehat{x}^k) \right] \right\| \\ &\leq \underbrace{\left\| \widehat{M}(\widehat{x}^k)^{-1} \left[\widehat{g}(\widehat{x}^\star) - \widehat{g}(\widehat{x}^k) - \widehat{H}(\widehat{x}^k)(\widehat{x}^\star - \widehat{x}^k) \right] \right\|}_{B_1} + \\ &\quad \underbrace{\left\| \left[I - \widehat{M}(\widehat{x}^k)^{-1} \widehat{H}(\widehat{x}^k) \right] (\widehat{x}^\star - \widehat{x}^k) \right\|}_{B_2} \end{aligned} \tag{D.100}$$

To bound B_1, it follows from (D.96) that,

$$B_1 \leq 2\beta \left\| \widehat{g}(\widehat{x}^\star) - \widehat{g}(\widehat{x}^k) - \widehat{H}(\widehat{x}^k)(\widehat{x}^\star - \widehat{x}^k) \right\|. \tag{D.101}$$

Furthermore, in the proof of [17, Theorem 6.3.2], it is shown that there exist $r_3, C_3 > 0$ such that,

$$\left\| \widehat{g}(\widehat{x}^\star) - \widehat{g}(\widehat{x}^k) - \widehat{H}(\widehat{x}^k)(\widehat{x}^\star - \widehat{x}^k) \right\| \leq C_3 \left\| \widehat{x}^k - \widehat{x}^\star \right\|^2, \tag{D.102}$$

for $x^k \in B_{r_3}(x^\star)$. Combining this result with (D.101), it holds that,

$$B_1 \leq 2\beta C_3 \left\| \widehat{x}^k - \widehat{x}^\star \right\|^2. \tag{D.103}$$

It remains to establish an upper bound for the matrix that appears in B_2:

$$I - \widehat{M}(\widehat{x}^k)^{-1} \widehat{H}(\widehat{x}^k) = \mathrm{D}\,\varphi(x) \circ (I - M(x^k)^{-1} \operatorname{Hess} f(x^k)) \circ (\mathrm{D}\,\varphi(x))^{-1}. \tag{D.104}$$

In the following, we first bound the norm of $I - M(x^k)^{-1} \operatorname{Hess} f(x^k)$. For any $\eta \in T_{x^k}\mathcal{M}$, let us consider the following quantity,

$$\begin{aligned} &\left\| \left(I - M(x^k)^{-1} \operatorname{Hess} f(x^k) \right) \eta \right\|_{H^k} \\ &= \left\| \left(\operatorname{Hess} f(x^k)^{-1} - M(x^k)^{-1} \right) \operatorname{Hess} f(x^k) \eta \right\|_{H^k}, \end{aligned} \tag{D.105}$$

where $\|\varepsilon\|_{H^k} = \sqrt{\langle \varepsilon, \operatorname{Hess} f(x^k)[\varepsilon] \rangle}$ denotes the norm induced by the Riemannian Hessian. Since $M(x^k) \approx_\epsilon \operatorname{Hess} f(x^k)$ by Assumption (A3), we can use Lemma D.4 to obtain an upper bound of (D.105),

$$\left\| \left(\operatorname{Hess} f(x^k)^{-1} - M(x^k)^{-1} \right) \operatorname{Hess} f(x^k) \eta \right\|_{H^k} \leq c(\epsilon) \|\eta\|_{H^k} . \tag{D.106}$$

In addition, using Assumption (A1), it holds that,

$$\sqrt{\mu_H} \|\varepsilon\| \leq \|\varepsilon\|_{H^k} \leq \sqrt{L_H} \|\varepsilon\| , \ \forall \varepsilon \in T_{x^k}\mathcal{M}. \tag{D.107}$$

Combining (D.105)-(D.107) yields,

$$\begin{aligned} \sqrt{\mu_H} \left\| \left(I - M(x^k)^{-1} \operatorname{Hess} f(x^k) \right) \eta \right\| &\leq \sqrt{L_H} c(\epsilon) \|\eta\| \\ \implies \left\| \left(I - M(x^k)^{-1} \operatorname{Hess} f(x^k) \right) \right\| &\leq \sqrt{\kappa_H} c(\epsilon). \end{aligned} \tag{D.108}$$

Combining (D.108) and (D.95) in (D.104), we conclude that,

$$B_2 \leq 2\sqrt{\kappa_H} c(\epsilon) \left\| \widehat{x}^k - \widehat{x}^\star \right\| . \tag{D.109}$$

Finishing the proof: We conclude the proof by combining the upper bounds (D.99), (D.103), and (D.109) in (D.94), and using the fact that $\left\| \widehat{x}^k - \widehat{x}^\star \right\| = \mathbf{d}(x^k, x^\star)$,

$$\mathbf{d}(x^{k+1}, x^\star) \leq 2\sqrt{\kappa_H} c(\epsilon) \, \mathbf{d}(x^k, x^\star) + (C_2 + 2\beta C_3) \, \mathbf{d}(x^k, x^\star)^2. \tag{D.110}$$

The linear convergence factor in (D.86) is obtained by noting that the second term on the right-hand side vanishes at a *quadratic rate*. □

D.4.2 Proof of Theorem 8.3

Proof. We will use the general linear convergence result established in Theorem D.1. However, in order to properly account for the gauge symmetry in rotation averaging, we need to invoke Theorem D.1 on the quotient manifold that underlies our

optimization problem. In the following, we break the proof into three main parts (highlighted in bold). The proof makes heavy use of results regarding Riemannian quotient manifolds. The reader is referred to [18, Chapter 9] for a comprehensive review.

Rotation Averaging and Optimization on Quotient Manifolds. Following standard references, we denote a Riemannian quotient manifold as $\mathcal{M} = \overline{\mathcal{M}}/\sim$. For rotation averaging (Problem 8.1), the *total space* is given by $\overline{\mathcal{M}} = \mathrm{SO}(d)^n$. Let $R = \{R_1, \ldots, R_n\}$ and $R' = \{R'_1, \ldots, R'_n\}$ be two points on $\overline{\mathcal{M}}$. We say that R and R' are equivalent if they are related via a left group action:

$$R \sim R' \iff \exists S \in \mathrm{SO}(d),\ SR_i = R'_i, \forall i = 1, \ldots, n. \tag{D.111}$$

The equivalence class represented by R is defined as,

$$[R] = \{(SR_1, \ldots, SR_n),\ S \in \mathrm{SO}(d)\}. \tag{D.112}$$

Note that the cost function of Problem 8.1 is invariant within an equivalence class. In the following, we use $T_R\overline{\mathcal{M}}$ to denote the usual tangent space at $R \in \overline{\mathcal{M}}$, and $T_{[R]}\mathcal{M}$ to denote the corresponding tangent space on the quotient manifold.

Given a retraction Retr on $\mathcal{M}$ (see [18, Chapter 9.6]), we can execute the Riemannian Newton's method on $\mathcal{M}$:

$$[R^{k+1}] = \mathrm{Retr}_{[R^k]}(\xi^k), \tag{D.113}$$

Above, $\xi^k \in T_{[R^k]}\mathcal{M}$ is the solution of the following linear equation,

$$\mathrm{Hess}\, f([R^k])[\xi^k] = -\,\mathrm{grad}\, f([R^k]), \tag{D.114}$$

where $\mathrm{grad}\, f([R^k])$ and $\mathrm{Hess}\, f([R^k])$ denote the Riemannian gradient and Hessian on the quotient manifold, respectively.

Expressing the iterates of Algorithm 8.4 on the quotient manifold. Recall that Algorithm 8.4 generates a sequence of iterates $\{R^k\}$ on the total space $\overline{\mathcal{M}}$. To prove convergence, we need to analyze the corresponding sequence of equivalence classes $\{[R^k]\}$ on the quotient manifold $\mathcal{M}$. Once we understand how $[R^k]$ evolves, we can prove the desired result by invoking Theorem D.1 on the quotient manifold $\mathcal{M}$. To begin with, we write the update step in Algorithm 8.4 in the following general form:

$$R^{k+1} = \overline{\mathrm{Retr}}_{R^k}(v^k), \tag{D.115}$$

where v^k is the vector corresponding to the matrix V^k in line 10; see (8.18) for how v^k and V^k are related. Together, $\overline{\mathrm{Retr}}_{R^k}(v^k)$ is the concise notation for the update steps in line 11-14, where we update each rotation R_i^k to $\mathrm{Exp}(v_i^k)R_i^k$. Our notation $\overline{\mathrm{Retr}}$ serves to emphasize that the retraction is performed on the total space $\overline{\mathcal{M}}$. Recall from (8.17)-(8.19) and Theorem 8.2 that v^k is a solution to the linear system,

$$(\widetilde{L} \otimes I_p)v^k = -\overline{g}(R^k), \tag{D.116}$$

where $\widetilde{L} \approx_\epsilon L \equiv L(G; w)$ is defined in (8.29) and $\overline{g}(R^k) \in T_R\overline{\mathcal{M}}$ is the Riemannian gradient in the total space defined in (8.12). In this proof, we will use the notations $\mathcal{N}_R$ and $\mathcal{H}_R$ to denote the vertical and horizontal spaces at $R \in \overline{\mathcal{M}}$.[1] By assumption, v^k is the unique solution to (D.116) that satisfies $v^k \perp \mathcal{N}_{R^k}$, *i.e.*, $v^k \in \mathcal{H}_{R^k}$. This implies that there is a unique tangent vector $\eta^k \in T_{[R^k]}\mathcal{M}$ on the tangent space of the quotient manifold such that v^k is the *horizontal lift* [18, Definition 9.25] of η^k at R^k:

$$v^k = \mathrm{lift}_{R^k}(\eta^k). \tag{D.117}$$

At any $R \in \overline{\mathcal{M}}$, define $\overline{M}(R) : \mathcal{H}_R \to \mathcal{H}_R$ to be the linear map,

$$\overline{M}(R) : v \mapsto (\widetilde{L} \otimes I_p)v, \tag{D.118}$$

[1]For rotation averaging, it turns out that definitions of vertical and horizontal spaces do not depend on the point R; *e.g.*, see (8.11) for the definition of the vertical space. However, in this proof we will still use the more general notations $\mathcal{N}_R$ and $\mathcal{H}_R$, which help us to emphasize that $\mathcal{N}_R$ and $\mathcal{H}_R$ are subspaces of the tangent space $T_R\overline{\mathcal{M}}$.

where $v \in \mathcal{H}_R$ is any vector from the horizontal space. Since $\ker(\widetilde{L} \otimes I_p) = \mathcal{N}_R$, it holds that $\overline{M}(R)$ is invertible on $\mathcal{H}_R$. Define $M([R]) : T_{[R]}\mathcal{M} \to T_{[R]}\mathcal{M}$ to be the corresponding linear map on the quotient manifold,

$$M([R]) = \text{lift}_R^{-1} \circ \overline{M}(R) \circ \text{lift}_R \,. \tag{D.119}$$

Note that $M([R])$ is indeed linear because when considered as a mapping $\text{lift}_R : T_{[R]}\mathcal{M} \to \mathcal{H}_R$, the horizontal lift is linear and invertible (see [18, Definition 9.25]). Furthermore, applying lift_R from the left on both sides of (D.119) shows that for any tangent vector $\eta \in T_{[R]}\mathcal{M}$,

$$\text{lift}_R \left(M([R])[\eta]\right) = \overline{M}(R)[\text{lift}_R(\eta)]. \tag{D.120}$$

By [18, Proposition 9.39], at iteration k, the Riemannian gradients on the total space and quotient space are related via,

$$\overline{g}(R^k) = \text{lift}_{R^k}(\text{grad}\, f([R^k])). \tag{D.121}$$

Combining (D.117)-(D.121), we see that (D.116) is equivalent to,

$$\text{lift}_{R^k}\left(M([R^k])[\eta^k]\right) = -\,\text{lift}_{R^k}(\text{grad}\, f([R^k])). \tag{D.122}$$

Applying $\text{lift}_{R^k}^{-1}$ to both sides of (D.122),

$$M([R^k])[\eta^k] = -\,\text{grad}\, f([R^k]). \tag{D.123}$$

In [18, Chapter 9.6], it is shown that $\text{Retr}_{[R]}(\eta) = [\overline{\text{Retr}}_R(\text{lift}_R(\eta))]$. Using this result, we see that the update equation on the total space (D.115) can be converted to the following update equation, defined on the quotient space:

$$[R^{k+1}] = [\overline{\text{Retr}}_{R^k}(v^k)] = \text{Retr}_{[R^k]}(\eta^k). \tag{D.124}$$

In summary, let $\{R_k\}$ denotes the iterates generated by Algorithm 8.4 on the total space $\overline{\mathcal{M}}$. We have shown that $\{R_k\}$ corresponds to a sequence $\{[R^k]\}$ on the quotient space $\mathcal{M}$ that evolves according to (D.123)-(D.124).

Invoking Theorem D.1 on the quotient manifold. To finish the proof, we will invoke Theorem D.1 to show that the sequence of iterates $[R^k]$ generated by (D.123)-(D.124) converges linearly to $[R^\star]$. This amounts to verifying that each condition in Assumption D.1 holds on the quotient manifold $\mathcal{M}$. To start, note that there exists $r' > 0$ such that for any $R \in \mathrm{SO}(d)^n$, the condition $\mathbf{d}([R], [R^\star]) < r'$ implies that $R \in B_r(R^\star)$ for some global minimizer $R^\star$ in the total space, where $B_r(R^\star)$ is the neighborhood within which Theorem 8.1 and Corollary 8.1 hold.

Verifying (A1). We need to derive lower and upper bounds for the Hessian of the quotient optimization problem $\operatorname{Hess} f([R])$. For any $\eta \in T_{[R]}\mathcal{M}$, let $v = \operatorname{lift}_R(\eta)$. By [18, Proposition 9.45], we have,

$$\langle \eta, \operatorname{Hess} f([R])\eta \rangle = \langle v, H(R)v \rangle \,, \tag{D.125}$$

where $H(R)$ is defined in (8.13). Since $v \in \mathcal{H}_R \equiv \mathcal{H}$ belongs to the horizontal space, we conclude using Corollary 8.1 that $\langle v, H(R)v \rangle \geq \mu_H \|v\|^2$ where μ_H is the constant defined in Corollary 8.1. Furthermore, since the quotient manifold $\mathcal{M}$ inherits the Riemannian metric from the total space $\overline{\mathcal{M}}$, we have $\|\eta\| = \|v\|$. We thus conclude that,

$$\langle \eta, \operatorname{Hess} f([R])\eta \rangle = \langle v, H(R)v \rangle \geq \mu_H \|v\|^2 = \mu_H \|\eta\|^2 . \tag{D.126}$$

Similarly, we can show that $\langle \eta, \operatorname{Hess} f([R])\eta \rangle \leq L_H \|\eta\|^2$ where L_H is also defined in Corollary 8.1. Therefore,

$$\mu_H I \preceq \operatorname{Hess} f([R]) \preceq L_H I. \tag{D.127}$$

Verifying (A2). We need to show that $M([R])$ defined in (D.119) is invertible and the operator norm of its inverse can be upper bounded. The invertibility follows

from (D.119) and the fact that both lift_R and $\overline{M}(R)$ are invertible on the horizontal space. To upper bound $M([R])^{-1}$, it is equivalent to derive a lower bound on $M([R])$. Let $\eta \in T_{[R]}\mathcal{M}$ and $v = \text{lift}_R(\eta)$. We have,

$$\langle \eta, M([R])\eta \rangle = \left\langle v, \overline{M}(R)v \right\rangle = v^\top (\widetilde{L} \otimes I_p)v \geq \lambda_2(\widetilde{L}) \left\| v \right\|^2. \tag{D.128}$$

The last inequality holds because $v \perp \mathcal{N}$. Thus, we conclude that,

$$\left\| M([R])^{-1} \right\| \leq 1/\lambda_2(\widetilde{L}). \tag{D.129}$$

Verifying (A3). Lastly, we need to show that the linear map $M([R])$ is a spectral approximation of the Riemannian Hessian $\text{Hess}\, f([R])$ on the quotient manifold. From Theorem 8.1, it holds that,

$$H(R) \approx_\delta L \otimes I_p. \tag{D.130}$$

In addition, from Theorem 8.2, we have

$$L \approx_\epsilon \widetilde{L} \implies L \otimes I_p \approx_\epsilon \widetilde{L} \otimes I_p. \tag{D.131}$$

Composing the two approximations yields,

$$H(R) \approx_{\delta+\epsilon} \widetilde{L} \otimes I_p. \tag{D.132}$$

Note that the above result directly implies the following approximation relation on the quotient manifold,

$$\text{Hess}\, f([R]) \approx_{\delta+\epsilon} M([R]). \tag{D.133}$$

To see this, note that for any $\eta \in T_{[R]}\mathcal{M}$ and $v = \text{lift}_R(\eta)$,

$$\begin{aligned} \langle \eta, \text{Hess}\, f([R])\eta \rangle &= \langle v, H(R)v \rangle \\ &\leq e^{\delta+\epsilon} \left\langle v, \overline{M}(R)v \right\rangle \\ &= e^{\delta+\epsilon} \langle \eta, M([R])\eta \rangle. \end{aligned} \tag{D.134}$$

The same argument leads to,

$$\langle \eta, \operatorname{Hess} f([R])\eta \rangle \geq e^{-\delta-\epsilon} \langle \eta, M([R])\eta \rangle . \tag{D.135}$$

□

D.4.3 Proof of Theorem 8.4

Proof. We prove this result using induction. The base case of $k = 1$ (first iteration) is true by Theorem 8.2. Now suppose (8.36) holds at iteration $k \geq 1$. Define $D^\star = M_t^\star - M_t^k$. By Theorem 8.2, the approximate refinement D^k computed at line 10 of Algorithm 8.5 satisfies,

$$\begin{aligned} \left\|D^k - D^\star\right\|_L \leq c(\epsilon) \left\|D^\star\right\|_L &\implies \left\|D^k + M_t^k - M_t^\star\right\|_L \leq c(\epsilon) \left\|M_t^k - M_t^\star\right\|_L \\ &\implies \left\|M_t^{k+1} - M_t^\star\right\|_L \leq c(\epsilon)^{k+1} \left\|M_t^\star\right\|_L . \end{aligned} \tag{D.136}$$

The second step above holds by the inductive hypothesis. □

D.5 Auxiliary Lemmas

Lemma D.4. *Let $L, \widetilde{L} \in \mathcal{S}_+^n$ such that $L \approx_\epsilon \widetilde{L}$. Let $B \in \mathbb{R}^{n\times p}$ be a matrix where each column of B lives in the image of L. Let $X^\star, \widetilde{X} \in \mathbb{R}^{n\times p}$ be matrices such that $LX^\star = B$ and $\widetilde{L}\widetilde{X} = B$. Then,*

$$\left\|X^\star - \widetilde{X}\right\|_L \leq c(\epsilon) \left\|X^\star\right\|_L , \tag{D.137}$$

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

Proof. We note that the proof of a similar result can be found at [142, Claim 2.4]. In the following, we provide the proof for the case where L and $\widetilde{L}$ are singular. The non-singular case can be proved in the same way by replacing matrix psuedoinverse

with the inverse. Observe that

$$\left\|X^\star - \widetilde{X}\right\|_L^2 = \sum_{i=1}^{p} \left\|X^\star_{[:,i]} - \widetilde{X}_{[:,i]}\right\|_L^2, \tag{D.138}$$

where $X^\star_{[:,i]}$ denotes the i-th column of $X^\star$. Therefore, we can first obtain an upper bound for the squared norm on a single column. To simplify notation, let $x^\star$ be a column of $X^\star$, and let $\widetilde{x}$ and b be the corresponding columns of $\widetilde{X}$ and B. Let us expand the squared norm,

$$\|x^\star - \widetilde{x}\|_L^2 = x^{\star\top} L x^\star - 2x^{\star\top} L\widetilde{x} + \widetilde{x}^\top L\widetilde{x}. \tag{D.139}$$

Note that since $L \approx_\epsilon \widetilde{L}$, we have $\ker(L) = \ker(\widetilde{L})$. In addition, any $\widetilde{x}$ where $\widetilde{L}\widetilde{x} = b$ can be written as $\widetilde{x} = \widetilde{L}^\dagger b + \widetilde{x}_\perp$ for some $\widetilde{x}_\perp \in \ker(L)$. Now, let us consider the middle term in (D.139),

$$x^{\star\top} L\widetilde{x} = x^{\star\top} L\widetilde{L}^\dagger b = x^{\star\top} L\widetilde{L}^\dagger L x^\star = (L^{1/2}x^\star)^\top (L^{1/2}\widetilde{L}^\dagger L^{1/2})(L^{1/2}x^\star). \tag{D.140}$$

The relation $\widetilde{L} \approx_\epsilon L$ implies $\widetilde{L}^\dagger \approx_\epsilon L^\dagger$, which is equivalent to,

$$e^{-\epsilon}\Pi \preceq L^{1/2}\widetilde{L}^\dagger L^{1/2} \preceq e^{\epsilon}\Pi, \tag{D.141}$$

where Π denotes the orthogonal projection onto $\text{image}(L^{1/2}\widetilde{L}^\dagger L^{1/2}) = \text{image}(L)$. By construction, it holds that $L^{1/2}x^\star \in \text{image}(L)$. Therefore,

$$x^{\star\top} L\widetilde{x} = (L^{1/2}x^\star)^\top (L^{1/2}\widetilde{L}^\dagger L^{1/2})(L^{1/2}x^\star) \geq e^{-\epsilon} \|x^\star\|_L^2 . \tag{D.142}$$

Next, expand the last term in (D.139):

$$\begin{aligned} \widetilde{x}^\top L\widetilde{x} &= b^\top \widetilde{L}^\dagger L\widetilde{L}^\dagger b = x^{\star\top} L\widetilde{L}^\dagger L\widetilde{L}^\dagger L x^\star \\ &= (L^{1/2}x^\star)^\top (L^{1/2}\widetilde{L}^\dagger L^{1/2})(L^{1/2}\widetilde{L}^\dagger L^{1/2})(L^{1/2}x^\star) \\ &= \left\|(L^{1/2}\widetilde{L}^\dagger L^{1/2})(L^{1/2}x^\star)\right\|_2^2 . \end{aligned} \tag{D.143}$$

Using (D.141), we conclude that,

$$\widetilde{x}^\top L \widetilde{x} \leq e^{2\epsilon} \left\| x^\star \right\|_L^2 . \tag{D.144}$$

Combining (D.142) and (D.144) in (D.139) yields,

$$\left\| x^\star - \widetilde{x} \right\|_L^2 \leq (1 + e^{2\epsilon} - 2e^{-\epsilon}) \left\| x^\star \right\|_L^2 = c(\epsilon)^2 \left\| x^\star \right\|_L^2 . \tag{D.145}$$

Finally, using this upper bound on (D.138) yields the desired result,

$$\begin{aligned} \left\| X^\star - \widetilde{X} \right\|_L^2 &= \sum_{i=1}^{p} \left\| X^\star_{[:,i]} - \widetilde{X}_{[:,i]} \right\|_L^2 \\ &\leq \sum_{i=1}^{p} c(\epsilon)^2 \left\| X^\star_{[:,i]} \right\|_L^2 \\ &= c(\epsilon)^2 \left\| X^\star \right\|_L^2 . \end{aligned} \tag{D.146}$$

□

Bibliography

[1] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular SLAM with multiple Micro Aerial Vehicles," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[2] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," in *Journal of Field Robotics (JFR)*, 2018.

[3] Y. Chang, K. Ebadi, C. E. Denniston, M. F. Ginting, A. Rosinol, A. Reinke, M. Palieri, J. Shi, A. Chatterjee, B. Morrell, *et al.*, "LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments," *IEEE Robotics and Automation Letters*, 2022.

[4] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, "COVINS: Visual-Inertial SLAM for Centralized Collaboration," in *IEEE International Symposium on Mixed and Augmented Reality Adjunct*, 2021.

[5] A. Cramariuc, L. Bernreiter, F. Tschopp, M. Fehr, V. Reijgwart, J. Nieto, R. Siegwart, and C. Cadena, "MAPLAB 2.0–A Modular and Multi-Modal Mapping Framework," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 520–527, 2022.

[6] Frank Dellaert et al., "Georgia Tech Smoothing And Mapping (GTSAM)." https://gtsam.org/, 2019.

[7] Sameer Agarwal et al., "Ceres Solver." http://ceres-solver.org.

[8] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: A general framework for (hyper) graph optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 9–13, 2011.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[10] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization," in *Proceedings of the Twenty Third International*

Conference on Artificial Intelligence and Statistics, vol. 108 of *Proceedings of Machine Learning Research*, pp. 2021–2031, PMLR, 26–28 Aug 2020.

[11] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[13] A. P. Eriksson, C. Olsson, F. Kahl, and T. Chin, "Rotation Averaging and Strong Duality," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] 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," *Intl. J. of Robotics Research*, vol. 38, no. 2-3, pp. 95–125, 2019.

[15] F. Dellaert, D. Rosen, J. Wu, R. Mahony, and L. Carlone, "Shonan Rotation Averaging: Global optimality by surfing $SO(p)^n$," in *European Conf. on Computer Vision (ECCV)*, 2020.

[16] H. Yang and L. Carlone, "Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 45, no. 3, pp. 2816–2834, 2022.

[17] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[18] N. Boumal, "An introduction to optimization on smooth manifolds," 2020.

[19] F. Dellaert, M. Kaess, *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[20] J. Briales and J. Gonzalez-Jimenez, "Cartan-Sync: Fast and Global SE(d)-Synchronization," *IEEE Robotics and Automation Letters*, 2017.

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

[22] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, "Asynchronous and Parallel Distributed Pose Graph Optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5819–5826, 2020.

[23] Y. Chang, Y. Tian, J. P. How, and L. Carlone, "Kimera-Multi: a System for Distributed Multi-Robot Metric-Semantic Simultaneous Localization and Mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 11210–11218, 2021.

[24] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems," *IEEE Trans. Robotics*, vol. 38, no. 4, 2022.

[25] Y. Tian, Y. Chang, L. Quang, A. Schang, C. Nieto-Granda, J. P. How, and L. Carlone, "Resilient and distributed multi-robot visual slam: Datasets, experiments, and lessons learned," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

[26] Y. Tian, A. S. Bedi, A. Koppel, M. Calvo-Fullana, D. M. Rosen, and J. P. How, "Distributed riemannian optimization with lazy communication for collaborative geometric estimation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4391–4398, IEEE, 2022.

[27] Y. Tian and J. P. How, "Spectral Sparsification for Communication-Efficient Collaborative Rotation and Translation Estimation." https://arxiv.org/pdf/2210.05020.pdf, 2022.

[28] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, "Graduated Non-Convexity for Robust Spatial Perception: From Non-Minimal Solvers to Global Outlier Rejection," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1127–1134, 2020.

[29] R. Tron, *Distributed optimization on manifolds for consensus algorithms and camera network localization*. The Johns Hopkins University, 2012.

[30] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation Averaging," *Intl. J. of Computer Vision*, 2013.

[31] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2015.

[32] S. Zhu, R. Zhang, L. Zhou, T. Shen, T. Fang, P. Tan, and L. Quan, "Very Large-Scale Global SfM by Distributed Motion Averaging," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[33] R. Tron and R. Vidal, "Distributed 3-D Localization of Camera Sensor Networks From 2-D Image Measurements," *IEEE Trans. on Automatic Control*, vol. 59, pp. 3325–3340, Dec 2014.

[34] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Trans. Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[35] T. Qin, S. Cao, J. Pan, and S. Shen, "A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors," *ArXiv*, vol. abs/1901.03642, 2019.

[36] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs," *Intl. J. of Robotics Research*, vol. 40, no. 12-14, pp. 1510–1546, 2021.

[37] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Trans. Robotics*, 2021.

[38] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *Intl. J. of Robotics Research*, 2017.

[39] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-Efficient Decentralized Visual SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 2466–2473, 2018.

[40] M. Giamou, K. Khosoussi, and J. P. How, "Talk resource-efficiently to me: Optimal communication planning for distributed loop closure detection," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3841–3848, IEEE, 2018.

[41] Y. Tian, K. Khosoussi, and J. P. How, "A Resource-Aware Approach to Collaborative Loop-Closure Detection with Provable Performance Guarantees," *Intl. J. of Robotics Research*, 2021.

[42] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," in *Vision Algorithms: Theory and Practice*, pp. 298–372, Springer, 2000.

[43] K. Ebadi, Y. Chang, M. Palieri, A. Stephens, A. Hatteland, E. Heiden, A. Thakur, N. Funabiki, B. Morrell, S. Wood, L. Carlone, and A.-a. Aghamohammadi, "LAMP: Large-Scale Autonomous Mapping and Positioning for Exploration of Perceptually-Degraded Subterranean Environments," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[44] P.-Y. Lajoie, B. Ramtoula, F. Wu, and G. Beltrame, "Towards collaborative simultaneous localization and mapping: a survey of the current research landscape," *Field Robotics*, 2021.

[45] J. Sivic and A. Zisserman, "Video Google: a text retrieval approach to object matching in videos," in *Intl. Conf. on Computer Vision (ICCV)*, 2003.

[46] D. Gálvez-López and J. D. Tardós, "Bags of Binary Words for Fast Place Recognition in Image Sequences," *IEEE Trans. Robotics*, vol. 28, pp. 1188–1197, October 2012.

[47] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5297–5307, 2016.

[48] D. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Intl. Conf. on Computer Vision (ICCV)*, pp. 1150–1157, 1999.

[49] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: speeded up robust features," in *European Conf. on Computer Vision (ECCV)*, 2006.

[50] T. Cieslewski and D. Scaramuzza, "Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 640–647, 2017.

[51] T. Cieslewski and D. Scaramuzza, "Efficient decentralized visual place recognition from full-image descriptors," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 78–82, 2017.

[52] Y. Tian, K. Khosoussi, M. Giamou, J. P. How, and J. Kelly, "Near-optimal budgeted data exchange for distributed loop closure detection," in *Robotics: Science and Systems (RSS)*, 2018.

[53] Y. Zhang, M. Hsiao, J. Dong, J. Engel, and F. Dellaert, "MR-iSAM2: Incremental Smoothing and Mapping with Multi-Root Bayes Tree for Multi-Robot SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 8671–8678, 2021.

[54] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[55] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using Constrained Factor Graphs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[56] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[57] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[58] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, vol. 23. Prentice hall Englewood Cliffs, NJ, 1989.

[59] R. Tron, J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "A Distributed Optimization Framework for Localization and Formation Control: Applications to Vision-Based Measurements," *IEEE Control Systems Magazine*, vol. 36, no. 4, pp. 22–44, 2016.

[60] J. Knuth and P. Barooah, "Collaborative 3D localization of robots from relative pose measurements using gradient descent on manifolds," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1101–1106, 2012.

[61] E. Cristofalo, E. Montijano, and M. Schwager, "GeoD: Consensus-based Geodesic Distributed Pose Graph Optimization," 2020.

[62] T. Fan and T. D. Murphey, "Generalized proximal methods for pose graph optimization," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, 2019.

[63] T. Fan and T. Murphey, "Majorization Minimization Methods for Distributed Pose Graph Optimization with Convergence Guarantees," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[64] T. Fan and T. Murphey, "Majorization Minimization Methods for Distributed Pose Graph Optimization," *arXiv preprint arXiv:2108.00083*, 2021.

[65] R. Murai, J. Ortiz, S. Saeedi, P. H. Kelly, and A. J. Davison, "A robot web for distributed many-device localisation," *arXiv preprint arXiv:2202.03314*, 2022.

[66] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3057–3064, 2011.

[67] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[68] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle Adjustment in the Large," in *European Conf. on Computer Vision (ECCV)* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 29–42, Springer Berlin Heidelberg, 2010.

[69] F. Dellaert, J. Carlson, V. Ila, K. Ni, and C. E. Thorpe, "Subgraph-preconditioned conjugate gradients for large scale SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2566–2571, 2010.

[70] A. Kushal and S. Agarwal, "Visibility Based Preconditioning for bundle adjustment," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1442–1449, 2012.

[71] A. Eriksson, J. Bastian, T. Chin, and M. Isaksson, "A Consensus-Based Framework for Distributed Bundle Adjustment," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[72] R. Zhang, S. Zhu, T. Fang, and L. Quan, "Distributed Very Large Scale Bundle Adjustment by Global Camera Consensus," in *Intl. Conf. on Computer Vision (ICCV)*, 2017.

[73] M. J. Schuster, K. Schmid, C. Brand, and M. Beetz, "Distributed stereo vision-based 6D localization and mapping for multi-robot teams," *Journal of Field Robotics (JFR)*, vol. 36, no. 2, pp. 305–332, 2019.

[74] G. S. Saeedi, M. Trentini, M. L. Seto, and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review," *Journal of Field Robotics (JFR)*, vol. 33, pp. 3–46, 2016.

[75] V. Tchuiev and V. Indelman, "Distributed Consistent Multi-Robot Semantic Localization and Mapping," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4649–4656, 2020.

[76] Y. Yue, C. Zhao, M. Wen, Z. Wu, and D. Wang, "Collaborative Semantic Perception and Relative Localization Based on Map Matching," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[77] M. Patel, M. Karrer, P. Bänninger, and M. Chli, "COVINS-G: A Generic Back-end for Collaborative Visual-Inertial SLAM," *arXiv preprint arXiv:2301.07147*, 2023.

[78] T. Zhang, L. Zhang, Y. Chen, and Y. Zhou, "CVIDS: A Collaborative Localization and Dense Mapping Framework for Multi-Agent Based Visual-Inertial SLAM," *IEEE Transactions on Image Processing*, 2022.

[79] H. Zhang, X. Chen, H. Lu, and J. Xiao, "Distributed and collaborative monocular simultaneous localization and mapping for multi-robot systems in large-scale environments," *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, p. 1729881418780178, 2018.

[80] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams," *IEEE Robotics and Automation Letters*, 2020.

[81] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise Consistent Measurement Set Maximization for Robust Multi-robot Map Merging," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Brisbane, Australia), pp. 1–8, May 2018.

[82] H. Xu, P. Liu, X. Chen, and S. Shen, "D^2 SLAM: Decentralized and Distributed Collaborative Visual-inertial SLAM System for Aerial Swarm," *arXiv preprint arXiv:2211.01538*, 2022.

[83] Y. Huang, T. Shan, F. Chen, and B. Englot, "DiSCo-SLAM: distributed scan context-enabled multi-robot lidar SLAM with two-stage global-local graph optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1150–1157, 2021.

[84] S. Zhong, Y. Qi, Z. Chen, J. Wu, H. Chen, and M. Liu, "DCL-SLAM: A Distributed Collaborative LiDAR SLAM Framework for a Robotic Swarm," *arXiv preprint arXiv:2210.11978*, 2022.

[85] P.-Y. Lajoie and G. Beltrame, "Swarm-SLAM: Sparse Decentralized Collaborative Simultaneous Localization and Mapping Framework for Multi-Robot Systems," *arXiv preprint arXiv:2301.06230*, 2023.

[86] K. J. Doherty, D. M. Rosen, and J. J. Leonard, "Spectral Measurement Sparsification for Pose-Graph SLAM," *arXiv preprint arXiv:2203.13897*, 2022.

[87] M. Tranzatto, F. Mascarich, L. Bernreiter, C. Godinho, M. Camurri, S. M. K. Khattak, T. Dang, V. Reijgwart, J. Loeje, D. Wisth, S. Zimmermann, H. Nguyen, M. Fehr, L. Solanka, R. Buchanan, M. Bjelonic, N. Khedekar, M. Valceschini, F. Jenelten, M. Dharmadhikari, T. Homberger, P. De Petris, L. Wellhausen, M. Kulkarni, T. Miki, S. Hirsch, M. Montenegro, C. Papachristos, F. Tresoldi, J. Carius, G. Valsecchi, J. Lee, K. Meyer, X. Wu, J. Nieto, A. Smith, M. Hutter, R. Siegwart, M. Mueller, M. Fallon, and K. Alexis, "CERBERUS: Autonomous Legged and Aerial Robotic Exploration in the Tunnel and Urban Circuits of the DARPA Subterranean Challenge," *Field Robotics*, 2021.

[88] Y. Zhu, Y. Kong, Y. Jie, S. Xu, and H. Cheng, "GRACO: A Multimodal Dataset for Ground and Aerial Cooperative Localization and Mapping," *IEEE Robotics and Automation Letters*, 2023.

[89] D. Feng, Y. Qi, S. Zhong, Z. Chen, Y. Jiao, Q. Chen, T. Jiang, and H. Chen, "S3E: A Large-scale Multimodal Dataset for Collaborative SLAM," *arXiv preprint arXiv:2210.13723*, 2022.

[90] J. Yin, A. Li, T. Li, W. Yu, and D. Zou, "M2DGR: A multi-sensor and multi-scenario slam dataset for ground robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2266–2273, 2021.

[91] L. Carlone, D. M. Rosen, G. Calafiore, J. J. Leonard, and F. Dellaert, "Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 125–132, 2015.

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

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

[94] P.-A. Absil, C. G. Baker, and K. A. Gallivan, "Trust-region methods on Riemannian manifolds," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 303–330, 2007.

[95] K. Khosoussi, S. Huang, and G. Dissanayake, "A Sparse Separable SLAM Back-End," *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1536–1549, 2016.

[96] A. Singer, "Angular synchronization by eigenvectors and semidefinite programming," *Applied and Computational Harmonic Analysis*, 2011.

[97] A. S. Bandeira, N. Boumal, and A. Singer, "Tightness of the maximum likelihood semidefinite relaxation for angular synchronization," *Mathematical Programming*, vol. 163, pp. 145–167, May 2017.

[98] A. Eriksson, C. Olsson, F. Kahl, and T. Chin, "Rotation Averaging with the Chordal Distance: Global Minimizers and Strong Duality," *IEEE Trans. Pattern Anal. Machine Intell.*, 2019.

[99] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[100] J. Neira and J. Tardós, "Data Association in Stochastic Mapping Using the Joint Compatibility Test," *IEEE Trans. Robot. Automat.*, vol. 17, no. 6, pp. 890–897, 2001.

[101] M. Bosse, G. Agamennoni, and I. Gilitschenski, "Robust Estimation and Applications in Robotics," *Foundations and Trends in Robotics*, vol. 4, no. 4, pp. 225–269, 2016.

[102] N. Sünderhauf and P. Protzel, "Switchable Constraints for Robust Pose Graph SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[103] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[104] R. Hartley, K. Aftab, and J. Trumpf, "L1 rotation averaging using the Weiszfeld algorithm," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3041–3048, IEEE, 2011.

[105] J. Casafranca, L. Paz, and P. Piniés, "A back-end ℓ_1 norm based solution for factor graph SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 17–23, 2013.

[106] A. Chatterjee and V. M. Govindu, "Efficient and Robust Large-Scale Rotation Averaging," in *Intl. Conf. on Computer Vision (ICCV)*, pp. 521–528, 2013.

[107] A. Chatterjee and V. M. Govindu, "Robust Relative Rotation Averaging," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 40, no. 4, pp. 958–972, 2018.

[108] G. Hu, K. Khosoussi, and S. Huang, "Towards a reliable SLAM back-end," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 37–43, 2013.

[109] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," in *Robotics: Science and Systems (RSS)*, July 2012.

[110] M. Pfingsthorn and A. Birk, "Simultaneous localization and mapping with multimodal probability distributions," *Intl. J. of Robotics Research*, vol. 32, no. 2, pp. 143–171, 2013.

[111] M. Pfingsthorn and A. Birk, "Generalized graph SLAM: Solving local and global ambiguities through multimodal and hyperedge constraints," *Intl. J. of Robotics Research*, vol. 35, no. 6, pp. 601–630, 2016.

[112] P. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling Perceptual Aliasing in SLAM via Discrete-Continuous Graphical Models," *IEEE Robotics and Automation Letters*, vol. 4, pp. 1232–1239, April 2019.

[113] L. Carlone and G. Calafiore, "Convex Relaxations for Pose Graph Optimization with Outliers," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1160–1167, 2018. arxiv preprint: 1801.02112.

[114] L. Carlone, A. Censi, and F. Dellaert, "Selecting good measurements via ℓ_1 relaxation: a convex approach for robust estimation over graphs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[115] M. Graham, J. How, and D. Gustafson, "Robust incremental SLAM with consistency-checking," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 117–124, Sept 2015.

[116] P. Antonante, V. Tzoumas, H. Yang, and L. Carlone, "Outlier-robust estimation: Hardness, minimally tuned algorithms, and applications," *IEEE Trans. Robotics*, vol. 38, no. 1, pp. 281–301, 2021.

[117] V. Indelman, E. Nelson, N. Michael, and F. Dellaert, "Multi-Robot Pose Graph Localization and Data Association from Unknown Initial Relative Poses via Expectation Maximization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[118] J. Dong, E. Nelson, V. Indelman, N. Michael, and F. Dellaert, "Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 5807–5814, 2015.

[119] N. Boumal, A. Singer, P.-A. Absil, and V. D. Blondel, "Cramér–Rao bounds for synchronization of rotations," *Information and Inference: A Journal of the IMA*, pp. 1–39, 2014.

[120] K. Khosoussi, M. Giamou, G. S. Sukhatme, S. Huang, G. Dissanayake, and J. P. How, "Reliable graphs for SLAM," *Intl. J. of Robotics Research*, 2019.

[121] Y. Chen, S. Huang, L. Zhao, and G. Dissanayake, "Cramér–Rao bounds and optimal design metrics for pose-graph SLAM," *IEEE Trans. Robotics*, 2021.

[122] J. A. Placed and J. A. Castellanos, "A General Relationship between Optimality Criteria and Connectivity Indices for Active Graph-SLAM," *IEEE Robotics and Automation Letters*, 2022.

[123] K. J. Doherty, D. M. Rosen, and J. J. Leonard, "Performance Guarantees for Spectral Initialization in Rotation Averaging and Pose-Graph SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 5608–5614, IEEE, 2022.

[124] L. Bernreiter, S. Khattak, L. Ott, R. Siegwart, M. Hutter, and C. Cadena, "Collaborative Robot Mapping using Spectral Graph Analysis," *arXiv preprint arXiv:2203.00308*, 2022.

[125] L. Carlone, "A convergence analysis for pose graph optimization via Gauss-Newton methods," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 965–972, 2013.

[126] K. Wilson, D. Bindel, and N. Snavely, "When is rotations averaging hard?," in *European Conf. on Computer Vision (ECCV)*, 2016.

[127] K. Wilson and D. Bindel, "On the distribution of minima in intrinsic-metric rotation averaging," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[128] S.-M. Nasiri, R. Hosseini, and H. Moradi, "Novel Parameterization for Gauss-Newton Methods in 3-D Pose Graph Optimization," *IEEE Trans. Robotics*, vol. 37, no. 3, pp. 780–797, 2021.

[129] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based SLAM," *Intl. J. of Robotics Research*, vol. 31, no. 11, pp. 1219–1230, 2012.

[130] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph SLAM," *IEEE Trans. Robotics*, vol. 30, no. 6, pp. 1371–1385, 2014.

[131] M. Mazuran, G. D. Tipaldi, L. Spinello, and W. Burgard, "Nonlinear Graph Sparsification for SLAM.," in *Robotics: Science and Systems (RSS)*, pp. 1–8, 2014.

[132] L. Paull, G. Huang, M. Seto, and J. J. Leonard, "Communication-constrained multi-AUV cooperative SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[133] M. Siami, A. Olshevsky, and A. Jadbabaie, "Deterministic and randomized actuator scheduling with guaranteed performance bounds," *IEEE Trans. on Automatic Control*, vol. 66, no. 4, pp. 1686–1701, 2020.

[134] N. K. Vishnoi, "Lx=b Laplacian Solvers and their Algorithmic Applications," *Foundations and Trends in Theoretical Computer Science*, 2013.

[135] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng, "Spectral sparsification of graphs: theory and algorithms," *Communications of the ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[136] D. A. Spielman and N. Srivastava, "Graph Sparsification by Effective Resistances," *SIAM Journal on Computing*, 2011.

[137] J. D. Batson, D. A. Spielman, and N. Srivastava, "Twice-ramanujan sparsifiers," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 255–262, 2009.

[138] R. Kyng and S. Sachdeva, "Approximate gaussian elimination for laplacians-fast, sparse, and simple," in *IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2016.

[139] D. Durfee, R. Kyng, J. Peebles, A. B. Rao, and S. Sachdeva, "Sampling random spanning trees faster than matrix multiplication," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 730–742, 2017.

[140] R. Peng and D. A. Spielman, "An efficient parallel solver for SDD linear systems," in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 333–342, 2014.

[141] Y. T. Lee, R. Peng, and D. A. Spielman, "Sparsified cholesky solvers for SDD linear systems," *arXiv preprint arXiv:1506.08204*, 2015.

[142] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 842–850, 2016.

[143] R. Tutunov, H. Bou-Ammar, and A. Jadbabaie, "Distributed Newton Method for Large-Scale Consensus Optimization," *IEEE Trans. on Automatic Control*, vol. 64, no. 10, pp. 3983–3994, 2019.

[144] S. J. Wright, "Coordinate Descent Algorithms," *Math. Program.*, vol. 151, pp. 3–34, June 2015.

[145] J. Nutini, I. Laradji, and M. Schmidt, "Let's Make Block Coordinate Descent Go Fast: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence," 2017.

[146] J. Mareček, P. Richtárik, and M. Takáč, "Distributed block coordinate descent for minimizing partially separable functions," in *Numerical Analysis and Optimization*, pp. 261–288, Springer, 2015.

[147] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[148] Y. Nesterov, "Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[149] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 4, pp. 3841–3846, IEEE, 2000.

[150] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Trans. Robotics*, vol. 21, no. 2, pp. 196–207, 2005.

[151] Z. Wen, D. Goldfarb, S. Ma, and K. Scheinberg, "Row by row methods for semidefinite programming," *Industrial Engineering*, pp. 1–21, 2009.

[152] P.-W. Wang, W.-C. Chang, and J. Z. Kolter, "The Mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints," *arXiv preprint arXiv:1706.00476*, 2017.

[153] M. A. Erdogdu, A. Ozdaglar, P. A. Parrilo, and N. D. Vanli, "Convergence Rate of Block-Coordinate Maximization Burer-Monteiro Method for Solving Large SDPs," *arXiv preprint arXiv:1807.04428*, 2018.

[154] Y. Tian, K. Khosoussi, and J. P. How, "Block-Coordinate Minimization for Large SDPs with Block-Diagonal Constraints," *arXiv preprint arXiv:1903.00597*, 2019.

[155] A. Agarwal and J. C. Duchi, "Distributed Delayed Stochastic Optimization," *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[156] F. Niu, B. Recht, C. Re, and S. Wright, "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent," *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[157] J. Liu and S. J. Wright, "Asynchronous Stochastic Coordinate Descent: Parallelism and Convergence Properties," *SIAM Journal on Optimization*, 2015.

[158] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An Asynchronous Parallel Stochastic Coordinate Descent Algorithm," *Journal of Machine Learning Research*, 2015.

[159] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization," *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[160] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous Decentralized Parallel Stochastic Gradient Descent," in *Intl. Conf. on Machine Learning (ICML)*, 2018.

[161] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari, "Asynchronous parallel algorithms for nonconvex optimization," *Mathematical Programming*, 2019.

[162] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Intl. Conf. on Machine Learning (ICML)*, pp. 3329–3337, PMLR, 2017.

[163] O. Shamir, N. Srebro, and T. Zhang, "Communication-Efficient Distributed Optimization Using an Approximate Newton-Type Method," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, JMLR.org, 2014.

[164] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily Aggregated Gradient for Communication-Efficient Distributed Learning," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 31, 2018.

[165] A. Javanmard, A. Montanari, and F. Ricci-Tersenghi, "Phase transitions in semidefinite relaxations.," *Proceedings of the National Academy of Sciences (PNAS)*, 2016.

[166] N. A. Lynch, *Distributed algorithms*. Elsevier, 1996.

[167] N. Boumal, P.-A. Absil, and C. Cartis, "Global rates of convergence for nonconvex optimization on manifolds," *IMA Journal of Numerical Analysis*, vol. 39, pp. 1–33, 02 2018.

[168] D. Rosen and L. Carlone, "Computational Enhancements for Certifiably Correct SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017. Workshop on "Introspective Methods for Reliable Autonomy".

[169] B. O'Donoghue and E. Candes, "Adaptive Restart for Accelerated Gradient Schemes," *Foundations of Computational Mathematics*, vol. 15, 04 2012.

[170] L. Barenboim and M. Elkin, "Distributed $(\Delta + 1)$-coloring in linear (in Δ) time," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 111–120, ACM, 2009.

[171] J. Schneider and R. Wattenhofer, "A new technique for distributed symmetry breaking," in *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pp. 257–266, ACM, 2010.

[172] D. M. Rosen, "Scalable Low-Rank Semidefinite Programming for Certifiably Correct Machine Perception," in *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, June 2020.

[173] L. Vandenberghe and S. Boyd, "Semidefinite Programming," *SIAM Review*, vol. 38, pp. 49–95, Mar. 1996.

[174] N. Boumal, V. Voroninski, and A. S. Bandeira, "The Non-convex Burer–Monteiro Approach Works on Smooth Semidefinite Programs," in *Advances in Neural Information Processing Systems (NIPS)*, (USA), pp. 2765–2773, Curran Associates Inc., 2016.

[175] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre, "Low-Rank Optimization on the Cone of Positive Semidefinite Matrices," *SIAM J. Optim.*, vol. 20, no. 5, pp. 2327–2351, 2010.

[176] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 3rd ed., 1996.

[177] Y. Saad, "On the Rates of Convergence of the Lanczos and the block-Lanczos Methods," *SIAM J. Numer. Anal.*, vol. 17, pp. 687–706, Oct. 1980.

[178] C. de Sa, B. He, I. Mitliagkas, C. Ré, and P. Xu, "Accelerated Stochastic Power Iteration," in *Proc. Mach. Learn. Res.*, pp. 58–67, 2018.

[179] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Sparse pose adjustment for 2d mapping," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[180] D. Martinec and T. Pajdla, "Robust Rotation and Translation Estimation in Multiview Reconstruction," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (Minneapolis, MN), pp. 1–8, June 2007.

[181] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab Toolbox for Optimization on Manifolds," *Journal of Machine Learning Research*, 2014.

[182] H. Tijms, *A First Course in Stochastic Models*. John Wiley and Sons, Ltd, 2004.

[183] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, 2006.

[184] W. Huang, P.-A. Absil, K. A. Gallivan, and P. Hand, "ROPTLIB: an object-oriented C++ library for optimization on Riemannian manifolds," tech. rep., Florida State University, 2016.

[185] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[186] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[187] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. García-Rodríguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," *ArXiv Preprint: 1704.06857*, 2017.

[188] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, "Parallel maximum clique algorithms with applications to network analysis," *SIAM Journal on Scientific Computing*, vol. 37, no. 5, pp. C589–C616, 2015.

[189] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *Intl. J. of Computer Vision*, 1996.

[190] P. Huber, *Robust Statistics*. John Wiley & Sons, New York, NY, 1981.

[191] Z. Zhang, "Parameter estimation techniques: a tutorial with application to conic fitting," *Image and Vision Computing*, vol. 15, no. 1, pp. 56–76, 1997.

[192] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," *ACM SIGGRAPH 2007 papers on - SIGGRAPH '07*, 2007.

[193] L. Carlone and A. Censi, "From Angular Manifolds to the Integer Lattice: Guaranteed Orientation Estimation With Application to Pose Graph Optimization," *IEEE Trans. Robotics*, vol. 30, no. 2, pp. 475–492, 2014.

[194] Army Research Laboratory, "Distributed and Collaborative Intelligent Systems and Technology Collaborative Research Alliance (DCIST CRA)." https://www.dcist.org/, 2020.

[195] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *Intl. J. of Robotics Research*, 2016.

[196] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, 1992.

[197] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847*, 2018.

[198] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[199] A. Reinke, M. Palieri, B. Morrell, Y. Chang, K. Ebadi, L. Carlone, and A. Agha-mohammadi, "LOCUS 2.0: Robust and Computationally Efficient LiDAR Odometry for Real-Time Underground 3D Mapping," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9043–9050, 2022.

[200] N. Hughes, Y. Chang, and L. Carlone, "Hydra: a real-time spatial perception engine for 3D scene graph construction and optimization," in *Robotics: Science and Systems (RSS)*, 2022.

[201] K. N. Chaudhury, Y. Khoo, and A. Singer, "Global Registration of Multiple Point Clouds Using Semidefinite Programming," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 468–501, 2015.

[202] D. P. O'Leary and B. W. Rust, "Variable projection for nonlinear least squares problems," *Computational Optimization and Applications*, vol. 54, pp. 579–593, 2013.

[203] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery.," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[204] C. Sweeney, "Theia Multiview Geometry Library: Tutorial & Reference." http://theia-sfm.org.

[205] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[206] K. Wilson and N. Snavely, "Robust Global Translations with 1DSfM," in *European Conf. on Computer Vision (ECCV)*, 2014.

[207] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3D," *ACM Trans. Graph.*, 2006.

[208] K. Ni and F. Dellaert, "Multi-level submap based SLAM using nested dissection," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2558–2565, 2010.

[209] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 273–278, 2010.

[210] E. Sucar, K. Wada, and A. Davison, "NodeSLAM: Neural Object Descriptors for Multi-View Shape Reconstruction," in *2020 International Conference on 3D Vision (3DV)*, pp. 949–958, 2020.

[211] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9107–9114, 2022.

[212] Y. Chang, N. Hughes, A. Ray, and L. Carlone, "Hydra-multi: Collaborative online construction of 3d scene graphs with multi-robot teams," *arXiv preprint arXiv:2304.13487*, 2023.

[213] M. Fernandez-Cortizas, H. Bavle, J. L. Sanchez-Lopez, P. Campoy, and H. Voos, "Multi s-graphs: A collaborative semantic slam architecture," *arXiv preprint arXiv:2305.03441*, 2023.

[214] J. Gallier, "The Schur Complement and Symmetric Positive Semidefinite (and Definite) Matrices." unpublished note, available online: http://www.cis.upenn.edu/~jean/schur-comp.pdf, Dec. 2010.

[215] A. Edelman, T. A. Arias, and S. T. Smith, "The Geometry of Algorithms with Orthogonality Constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, p. 303–353, Apr. 1999.

[216] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[217] D. A. Spielman, "Spectral and Algebraic Graph Theory." http://cs-www.cs.yale.edu/homes/spielman/sagt/sagt.pdf, 2019.

[218] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.