NORTHWESTERN UNIVERSITY

Efficient and Guaranteed Geometric Methods for Motion Generation and Perception

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Mechanical Engineering

By

Taosha Fan

EVANSTON, ILLINOIS

March 2022

ABSTRACT

Efficient and Guaranteed Geometric Methods for Motion Generation and Perception

Taosha Fan

Even though a number of techniques have been developed for motion generation and perception, few of them focus on the computational efficiency and theoretical guarantees at the same time. Typically, improved guarantees come with increased complexity, making theoretically guaranteed methods challenging use in real-time applications. Thus, existing methods usually have to ignore either efficiency or guarantees in practical implementation. Nevertheless, numerous problems in motion generation and perception require computational efficiency as well as theoretical guarantees, making the implementation of existing techniques strictly limited. To address this issue, we present efficient and guaranteed methods for motion generation and perception by utilizing geometry and optimization. In this thesis, we develop fast algorithms for higher-order variational integrators with linear- and quadratic-time complexity for integration and linearization, respectively; we make use of the complex number representation to solve the planar graph-based SLAM that is not only certifiably correct but also more efficient and robust; we propose majorization

minimization methods for distributed pose graph optimization that have provable convergence to first-order critical points and can be accelerated with no loss of theoretical guarantees; we present a sparse constrained formulation for 3D human pose and shape estimation with which a linear-time algorithm is derived to compute the Gauss-Newton direction and the optimization time is reduced from tens of seconds to several milliseconds. In spite of the theoretical guarantees, all of these aforementioned methods achieve the state-of-the-art performances in terms of both accuracy and efficiency for their specific applications in motion generation and perception.

Acknowledgements

First of all, I would like to thank my advisor Prof. Todd Murphey for his kind support and guidance. Todd has given me tremendous freedom to explore my research interests even though some of them sound senseless at the beginning. Whenever I encounter difficulties, Todd always makes time for me and tries his best to help. I can never express my thanks enough. I'm also grateful to Prof. Randy Freeman and Prof. Brenna Argall for being my committee members and giving valuable and inspiring feedback on my dissertation. I further want to thank my colleagues and friends in Northwestern with whom I spent lots of wonderful time these years. At last, I want to thank my family who has provided me with endless support and care throughout my life. This dissertation is impossible without them.

Table of Contents

List of Tables

is the number of measurements for all the body parts, and N_i is the number of measurements associated with body part i. 303

List of Figures

CHAPTER 1

Introduction

Motion generation and perception are essential and have broad applications in robotics. Despite that a number of techniques have been proposed, most of them have to make a compromise between computational efficiency and theoretical guarantees, and thus, their implementation is strictly limited in practice. To address these issues, this thesis focuses on efficient and guaranteed methods for motion generation and perception. In particular, our methods are developed from a geometric perspective and we show that numerous problems in motion generation and perception can be reasonably formulated and solved by utilizing geometry and optimization.

1.1. Contributions

1.1.1. Higher-Order Variational Integrators

Numerical integrators are critical to the motion generation for robots. Due to the preservation of mechanical quantities, Variational Integrators (VI) are well-known for their longer-time stability and high accuracy [**15**]. In spite of this, variational integrators are time-consuming for computation, and thus, difficult to be implemented in real time. In Chapter 2, we present algorithms that significantly improve the computational efficiency of higher-order variational integrators. Our algorithms are applicable to variational integrators of arbitrarily high order, and more importantly, reduce the complexity from $O(n^3)$ to $O(n)$ for integration and $O(n^4)$ to $O(n^2)$ for linearization—n being the number

of joints. These improvements make higher-order variational integrators well suited for the simulation and trajectory optimization of complex robotic systems.

1.1.2. Planar Graph-Based SLAM

When navigating without GPS, a robot is required to estimate its location as well as build the map of the environment. Such a problem is termed as Simultaneous Localization and Mapping (SLAM) [**16**]. In general, SLAM problems are formulated as a graph where the vertices are either robot's locations or landmark's positions while the edges are the available noisy measurements [**17**]. In Chapter 3, we present CPL-SLAM that formulates planar SLAM using the complex number representation. Even though the resulting optimization problem is nonconvex, CPL-SLAM is certifiably correct and guaranteed to recover the globally optimal solution regardless of the initialization as long as the measurement noise is under a certain threshold. In addition, as a result of the complex number representation, CPL-SLAM is faster and more robust than the other certifiably correct algorithms for SLAM [**7**].

1.1.3. Distributed Pose Graph Optimization

Pose Graph Optimization (PGO) has extensive applications in autonomous driving, AR/VR, mapping, etc. [**18**]. Even though centralized PGO has been well studied, it can not solve large-scale problems due to the limitation of computational resources. In contrast, distributed PGO has no such restrictions and applies to problems of all the scales. Since the communication latency has been greatly reduced, distributed PGO is more concerned with the rates and guarantees of the convergence. In Chapter 4, we present

major minimization methods for distributed PGO that have provable convergence to first-order critical points under mild conditions. Furthermore, we exploit Nesterov's method and adaptive restarts to accelerate the convergence of distributed PGO without sacrificing any theoretical guarantees. Last but not the least, our majorization minimization methods for distributed PGO can be fully decentralized while achieving comparable performance to these with a master node to communicate with all the nodes in the network.

1.1.4. 3D Human Pose and Shape Estimation

Estimating 3D human poses and shapes from images are widely used in embodied AI, robotics, AR/VR. We can solve this problem by either optimization or regression methods. Although more popular, regression methods depend on optimization methods for neural network training and output refining. Therefore, optimization methods remain important for 3D human pose and shape estimation. One of the most major drawbacks for optimization methods is that they suffer from high computation times. This mainly results from the inefficiency to compute the Gauss-Newton direction when solving the optimization problem. In Chapter 5, we present a sparse constrained formulation for 3D human pose and shape estimation that is equivalent to existing optimization methods under mild conditions. Furthermore, we exploit the underlying sparsity and constraints of our formulation and derive algorithms that have the computation of the Gauss-Newton direction scale linearly with the number of joints. In contrast, existing optimization methods have the cubic complexity. As a result of the sparse constrained formulation, our methods reduce the optimization from tens of seconds to less than 4 milliseconds with no loss of accuracy, which is orders of magnitude faster.

CHAPTER 2

Efficient Computation of Higher-Order Variational Integrators

This chapter addresses the problem of efficiently computing higher-order variational integrators in simulation and trajectory optimization of mechanical systems as those often found in robotic applications. We develop $O(n)$ algorithms to evaluate the discrete Euler-Lagrange (DEL) equations and compute the Newton direction for solving the DEL equations, which results in linear-time variational integrators of arbitrarily high order. To our knowledge, no linear-time higher-order variational or even implicit integrators have been developed before. Moreover, an $O(n^2)$ algorithm to linearize the DEL equations is presented, which is useful for trajectory optimization. These proposed algorithms eliminate the bottleneck of implementing higher-order variational integrators in simulation and trajectory optimization of complex robotic systems. The efficacy of this chapter is validated through comparison with existing methods, and implementation on various robotic systems—including trajectory optimization of the Spring Flamingo robot, the LittleDog robot and the Atlas robot. The results illustrate that the same integrator can be used for simulation and trajectory optimization in robotics, preserving mechanical properties while achieving good scalability and accuracy.

2.1. Introduction

Variational integrators conserve symplectic form, constraints and energetic quantities [**15**, **19**–**23**]. As a result, variational integrators generally outperform the other types

of integrators with respect to numerical accuracy and stability, thus permitting large time steps in simulation and trajectory optimization, which is useful for complex robotic systems [**15**,**19**–**23**]. Moreover, variational integrators can also be regularized for collisions and friction by leveraging the linear complementarity problem (LCP) formulation [**24**,**25**].

The computation of variational integrators is comprised of the discrete Euler-Lagrange equation (DEL) evaluation, the descent direction computation for solving the DEL equations and the DEL equation linearization. The computation of these three phases of variational integrators can be accomplished with automatic differentiation and our prior methods [**19**, **21**], both of which are $O(n^2)$ to evaluate the DEL equations and $O(n^3)$ to compute the Newton direction and linearize the DEL equations for an n-degree-of-freedom mechanical system. Recently, a linear-time second-order variational integrator was developed in [**4**], which uses the quasi-Newton method and works for small time steps and comparatively simple mechanical systems.

Higher-order variational integrators are needed for greater accuracy in predicting the dynamic motion of robots [**26**, **27**]. However, the computation of higher-order variational integrators has rarely been addressed. The quasi-Newton method in [**4**] only applies to second-order variational integrators, and while automatic differentiation and our prior methods [**19**, **21**] are implementable for higher-order variational integrators, the complexity increases superlinearly as the integrator order increases.

This chapter is built upon the preliminary results in [**28**]. We address the computation efficiency of higher-order variational integrators and develop: i) an $O(n)$ method for the evaluation of the DEL equations, ii) an $O(n)$ method for the computation of the Newton direction, and iii) an $O(n^2)$ method for the linearization of the DEL equations.

The proposed characteristics i) – iii) eliminate the bottleneck of implementing higher-order variational integrators in simulation and trajectory optimization of complex robotic systems, and to the best of our knowledge, no similar work has been presented before. In particular, we believe that the resulting variational integrator from i) and ii) is the first exactly linear-time implicit integrator of third or higher order for mechanical systems.

The rest of this chapter is organized as follows. Section 2.2 reviews higher-order variational integrators, the Lie group formulation of rigid body motion and the tree representation of mechanical systems. Sections 2.3 and 2.4 respectively detail the linear-time higher-order variational integrator and the quadratic-time linearization, which are the main contributions of this chapter. Section 2.5 compares our work with existing methods, and Section 2.6 presents examples of trajectory optimization for the Spring Flamingo robot, the LittleDog robot and the Atlas robot. The conclusions are made in Section 2.7.

2.2. Preliminaries and Notation

In this section, we review higher-order variational integrators, the Lie group formulation of rigid body motion, and the tree representation of mechanical systems. In addition, notation used throughout this chapter is introduced accordingly.

2.2.1. Higher-Order Variational Integrators

In this chapter, higher-order variational integrators are derived with the methods in [**15**, **29**, **30**].

A trajectory $(q(t),\, \dot{q}(t))$ where $0 \leq t \leq T$ of a forced mechanical system should satisfy the *Lagrange-d'Alembert principle*:

$$\delta\mathfrak{S} = \delta \mathrm{int}_0^T \mathcal{L}(q, \dot{q})dt + \mathrm{int}_0^T \mathcal{F}(t) \cdot \delta q dt = 0 \tag{2.1}$$

in which $\mathcal{L}(q, \dot{q})$ is the system's Lagrangian and $\mathcal{F}(t)$ is the generalized force. Provided that the time interval $[0,\, T]$ is evenly divided into N sub-intervals with $\Delta t = T/N$, and each $q(t)$ over $[k\Delta t,\, (k+1)\Delta t]$ is interpolated with $s+1$ control points $q^{k,\alpha} = q(t^{k,\alpha})$ in which $\alpha = 0,\, 1,\, \cdots,\, s$ and $k\Delta t = t^{k,0} < t^{k,1} < \cdots < t^{k,s} = (k+1)\Delta t$, then there are coefficients $b^{\alpha\beta}$ $(0 \leq \alpha,\, \beta \leq s)$ such that

$$\dot{q}(t^{k,\alpha}) \approx \dot{q}^{k,\alpha} = \frac{1}{\Delta t} \sum_{\beta=0}^{s} b^{\alpha\beta} q^{k,\beta}. \tag{2.2}$$

In this chapter, we assume that the quadrature points of the quadrature rule are also $t^{k,\alpha}$ though our algorithms in Sections 2.3 and 2.4 can be generalized for any quadrature rules. Then the Lagrange-d'Alembert principle Eq. (2.1) is approximated as

$$\delta\mathfrak{S} \approx \sum_{k=0}^{N-1} \sum_{\alpha=0}^{s} w^{\alpha} \left[\delta\mathcal{L}(q^{k,\alpha}, \dot{q}^{k,\alpha}) + \mathcal{F}(t^{k,\alpha}) \cdot \delta q^{k,\alpha} \right] \cdot \Delta t = 0, \tag{2.3}$$

where w^{α} are weights of the quadrature rule used for integration. In variational integrators, the *discrete Lagrangian* and the *discrete generalized force* are defined to be

$$\mathcal{L}_d(q^{k,0},\, q^{k,1},\, \cdots,\, q^{k,s}) = \sum_{\alpha=0}^{s} w^{\alpha} \mathcal{L}(q^{k,\alpha}, \dot{q}^{k,\alpha}) \Delta t \tag{2.4}$$

and $\mathcal{F}_d^{k,\alpha}(t^{k,\alpha}) = w^{\alpha} \mathcal{F}(t^{k,\alpha}) \Delta t$, respectively. Note that by definition we have $t^{k,s} = t^{k+1,0}$ and $q^{k,s} = q^{k+1,0}$, and as a result of Eq. (2.3), we obtain

$$p^k + \mathbb{D}_1 \mathcal{L}_d(\overline{q}^k) + \mathcal{F}_d^{k,0} = 0, \tag{2.5a}$$

$$\mathbb{D}_{\alpha+1} \mathcal{L}_d(\overline{q}^k) + \mathcal{F}_d^{k,\alpha} = 0 \quad \forall \alpha = 1,\, \cdots,\, s-1, \tag{2.5b}$$

$$p^{k+1} = \mathbb{D}_{s+1}\mathcal{L}_d(\overline{q}^k) + \mathcal{F}_d^{k,s} \tag{2.5c}$$

in which p^k is the *discrete momentum*, $\overline{q}^k$ stands for the tuple $(q^{k,0}, q^{k,1}, \cdots, q^{k,\alpha})$, and $\mathbb{D}_{\alpha+1}\mathcal{L}_d$ is the derivative with respect to $q^{k,\alpha}$. Note that Eq. (2.5) is known as the *discrete Euler-Lagrangian (DEL) equations*, which implicitly define an update rule $(q^{k,0}, p^k) \to (q^{k+1,0}, p^{k+1})$ by solving sn nonlinear equations from Eqs. (2.5a) and (2.5b). In a similar way, for mechanical systems with constraints $h(q, \dot{q}) = 0$, we have

$$p^k + \mathbb{D}_1\mathcal{L}_d(\overline{q}^k) + \mathcal{F}_d^{k,0} + A^{k,0}(q^{k,0}) \cdot \lambda^{k,0} = 0, \tag{2.6a}$$

$$\mathbb{D}_{\alpha+1}\mathcal{L}_d(\overline{q}^k) + \mathcal{F}_d^{k,\alpha} + A^{k,\alpha}(q^{k,\alpha}) \cdot \lambda^{k,\alpha} = 0 \quad \forall \alpha = 1, \cdots, s-1, \tag{2.6b}$$

$$p^{k+1} = \mathbb{D}_{s+1}\mathcal{L}_d(\overline{q}^k) + \mathcal{F}_d^{k,s}, \tag{2.6c}$$

$$h^{k,\alpha}(q^{k+1,\alpha}, \dot{q}^{k+1,\alpha}) = 0 \quad \forall \alpha = 1, \cdots, s \tag{2.6d}$$

in which $A^{k,\alpha}(q^{k,\alpha})$ is the discrete constraint force matrix and $\lambda^{k,\alpha}$ is the discrete constraint force.

The resulting higher-order variational integrator is referred as the Galerkin integrator [**15**, **29**, **30**], the accuracy of which depends on the number of control points as well as the numerical quadrature of the discrete Lagrangian. If there are $s+1$ control points and the Lobatto quadrature is employed, then the resulting variational integrator has an accuracy of order $2s$ [**29**,**30**]. The Galerkin integrator includes the *trapezoidal variational*

integrator and the *Simpson variational integrator* as shown in Examples 2.1 and 2.2, the DEL equations of which are given by Eqs. (2.5) and (2.6).

Example 2.1. The trapezoidal variational integrator is a second-order integrator with two control points $\overline{q}^k = (q^{k,0}, q^{k,1})$ such that $q^{k,0} = q\big(k\Delta t\big)$ and $q^{k,1} = q\big((k+1)\Delta t\big)$, $\dot{q}^{k,0} = \dot{q}^{k,1} = \frac{q^{k,1}-q^{k,0}}{\Delta t}$, and $\mathcal{L}_d(\overline{q}^k) = \frac{\Delta t}{2}\left[\mathcal{L}(q^{k,0}, \dot{q}^{k,0}) + \mathcal{L}(q^{k,1}, \dot{q}^{k,1})\right]$.

Example 2.2. The Simpson variational integrator is a fourth-order integrator with three control points $\overline{q}^k = (q^{k,0}, q^{k,1}, q^{k,2})$ in which $q^{k,0} = q\big(k\Delta t\big)$, $q^{k,0} = q\big((k+\frac{1}{2})\Delta t\big)$ and $q^{k,2} = q\big((k+1)\Delta t\big)$, $\dot{q}^{k,0} = \frac{4q^{k,1}-3q^{k,0}-q^{k,2}}{\Delta t}$, $\dot{q}^{k,1} = \frac{q^{k,2}-q^{k,0}}{\Delta t}$ and $\dot{q}^{k,2} = \frac{q^{k,0}+3q^{k,2}-4q^{k,1}}{\Delta t}$, and $\mathcal{L}_d(\overline{q}^k) = \frac{\Delta t}{6}\left[\mathcal{L}(q^{k,0}, \dot{q}^{k,0}) + 4\mathcal{L}(q^{k,1}, \dot{q}^{k,1}) + \mathcal{L}(q^{k,2}, \dot{q}^{k,2})\right]$.

2.2.2. The Lie Group Formulation of Rigid Body Motion

The configuration of a rigid body $g = (R, p) \in SE(3)$ can be represented as a 4×4 matrix $g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$ in which $R \in SO(3)$ is a rotation matrix and $p \in \mathbb{R}^3$ is a position vector. The body velocity of the rigid body $v = (\omega, v_O) \in T_eSE(3)$ is an element of the Lie algebra and can be represented either as a 6×1 vector $v = (g^{-1}\dot{g})^\vee = \begin{bmatrix} \omega^T & v_O^T \end{bmatrix}$ or a 4×4 matrix $\hat{v} = g^{-1}\dot{g} = \begin{bmatrix} \hat{\omega} & v_O \\ 0 & 0 \end{bmatrix}$ in which $\omega = (\omega_x, \omega_y, \omega_z) \in T_eSO(3)$ is the angular velocity, v_O is the linear velocity, $\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}$, and the hat "$\wedge$" and unhat"$\vee$" are linear operators that relate the vector and matrix representations. The same representation and operators also apply to the spatial velocity $\overline{v} = (\overline{\omega}, \overline{v}_O) \in T_eSE(3)$,

whose 6×1 vector and 4×4 matrix representations are respectively $\overline{v} = (\dot{g}g^{-1})^{\vee}$ and $\hat{\overline{v}} = \dot{g}g^{-1}$.

In the rest of this chapter, if not specified, vector representation is used for $T_eSE(3)$, such as v, $\overline{v}$, etc., and the adjoint operators Ad_g and $\mathrm{ad}_v : T_eSE(3) \to T_eSE(3)$ can be accordingly represented as 6×6 matrices $\mathrm{Ad}_g = \begin{bmatrix} R & 0 \\ \hat{p}R & R \end{bmatrix}$ and $\mathrm{ad}_v = \begin{bmatrix} \hat{\omega} & 0 \\ \hat{v}_O & \hat{\omega} \end{bmatrix}$ such that $\overline{v} = \mathrm{Ad}_g v$ and $\mathrm{ad}_{v_1} v_2 = (\hat{v}_1\hat{v}_2 - \hat{v}_2\hat{v}_1)^{\vee}$. For consistence, the dual Lie algebra $T_e^*SE(3)$ uses the 6×1 vector representation as well. As a result, the body wrench $F = (\tau, f_O) \in T_e^*SE(3)$ is represented as a 6×1 vector $F = \begin{bmatrix} \tau^T & f_O^T \end{bmatrix}^T$ in which $\tau \in T_e^*SO(3)$ is the torque and f_O is the linear force so that $\langle F, v \rangle = F^T v$. Moreover, we define the linear operator $\mathrm{ad}_F^D : T_eSE(3) \to T_e^*SE(3)$ which is represented as a 6×6 matrix $\mathrm{ad}_F^D = \begin{bmatrix} \hat{\tau} & \hat{f}_O \\ \hat{f}_O & 0 \end{bmatrix}$ so that $F^T \mathrm{ad}_{v_1} v_2 = v_2^T \mathrm{ad}_F^D v_1 = -v_1^T \mathrm{ad}_F^D v_2$ for $v_1, v_2 \in T_eSE(3)$. The same representation and operators also apply to the spatial wrench $\overline{F} = \mathrm{Ad}_g^{-T} F = (\overline{\tau}, \overline{f}_O)$ which is paired with the spatial velocity $\overline{v} = \mathrm{Ad}_g v$.

2.2.3. The Tree Representation of Mechanical Systems

In general, a mechanical system with n inter-connected rigid bodies indexed as $1, 2, \cdots, n$ can be represented through a tree structure so that each rigid body has a single parent and zero or more children [**19**, **31**], and such a representation is termed as *tree representation*. In this chapter, the spatial frame is denoted as $\{0\}$, which is the root of the tree representation, and we denote the body frame of rigid body i as $\{i\}$, and the parent, ancestors, children and descendants of rigid body i as $\mathrm{par}(i)$, $\mathrm{anc}(i)$, $\mathrm{chd}(i)$ and $\mathrm{des}(i)$, respectively. Since all joints can be modeled using a combination of revolute joints and

prismatic joints, we assume that each rigid body i is connected to its parent by a one-degree-of-freedom joint i which is either a revolute or a prismatic joint and parameterized by a real scalar $q_i \in \mathbb{R}$. As a result, the tree representation is parameterized with n generalized coordinates $q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}^T \in \mathbb{R}^n$. For each joint i, the joint twist with respect to frame $\{0\}$ and $\{i\}$ are respectively denoted as 6×1 vectors $\overline{S}_i = \begin{bmatrix} \overline{s}_i^T & \overline{n}_i^T \end{bmatrix}^T$ and $S_i = \begin{bmatrix} s_i^T & n_i^T \end{bmatrix}^T$ in which $\overline{s}_i$, s_i are 3×1 vectors corresponding to rotation and $\overline{n}_i$, n_i are 3×1 vectors corresponding to translation. Note that S_i, s_i and n_i are constant by definition. Moreover, $\overline{S}_i$ and S_i are related as $\overline{S}_i = \mathrm{Ad}_{g_i} S_i$ where $g_i \in SE(3)$ is the configuration of rigid body i, and $\dot{\overline{S}}_i = \mathrm{ad}_{\overline{v}_i} \overline{S}_i$, where $\overline{v}_i \in T_e SE(3)$ is the spatial velocity of rigid body i.

It is assumed without loss of generality in this chapter that the origin of frame $\{i\}$ is the mass center of rigid body i, and $j \in \mathrm{des}(i)$ only if $i < j$, or equivalently $j \in \mathrm{anc}(i)$ only if $i > j$.

The rigid body dynamics can be computed through the tree representation. The configuration $g_i = (R_i, p_i) \in SE(3)$ of rigid body i is $g_i = g_{\mathrm{par}(i)} g_{\mathrm{par}(i),i}(q_i)$ in which $g_{\mathrm{par}(i),i}(q_i) = g_{\mathrm{par}(i),i}(0) \exp(\hat{S}_i q_i)$ is the rigid body transformation from frame $\{i\}$ to its parent frame $\{\mathrm{par}(i)\}$, and the spatial velocity $\overline{v}_i$ of rigid body i is $\overline{v}_i = \overline{v}_{\mathrm{par}(i)} + \overline{S}_i \cdot \dot{q}_i$. In addition, the spatial inertia matrix $\overline{M}_i$ of rigid body $\{i\}$ with respect to frame $\{0\}$ is $\overline{M}_i = \mathrm{Ad}_{g_i}^{-T} M_i \mathrm{Ad}_{g_i}^{-1}$ in which $M_i = \mathrm{diag}\{\mathcal{I}_i, m_i \mathbf{I}\} \in \mathbb{R}^{6 \times 6}$ is the constant body inertia matrix of rigid body i, $\mathcal{I}_i \in \mathbb{R}^{3 \times 3}$ is the body rotational inertia matrix, $m_i \in \mathbb{R}$ is the mass and $\mathbf{I} \in \mathbb{R}^{3 \times 3}$ is the identity matrix.

In rigid body dynamics, an important notion is the *articulated body* [**31**]. In terms of the tree representation, articulated rigid body i consists of rigid body i and all its descendants $j \in \mathrm{des}(i)$, and the interactions with articulated body i can only be made through rigid body i, which is known as the handle of the articulated body i.

In the last thirty years, a number of algorithms for efficiently computing the rigid body dynamics have been developed based on tree representations and articulated bodies [**31**–**33**], making explicit integrators have $O(n)$ complexity for an n-degree-of-freed-om mechanical system. Even though the same algorithms might be used for the evaluation of implicit integrators, *none of them can be used for the computation of the Newton direction for solving implicit integrators*. If the residue is r^k, the Newton direction of an implicit integrator is computed as $\delta q^k = -\mathcal{J}(q^k)^{-1} r^k$; however, the Jacobian matrix $\mathcal{J}(q^k)$ is usually asymmetric and indefinite, and has a size greater than $n \times n$ for higher-order implicit integrators, which means that the computation of implicit integrators is distinct from explicit integrators whose computation is simply a combination of the algorithms in [**31**–**33**] with an appropriate integration scheme. Furthermore, the computation of implicit integrators is much more complicated than the computation of forward and inverse dynamics and out of the scope of those algorithms in [**31**–**33**].

2.2.4. Recursive Computation of the Variations and Derivatives

In addition to the computation of rigid body dynamics as those in Section 2.2.3, the tree representation can also be used to compute the variations and derivatives.

In the tree representation, the configuration $g_i \in SE(3)$ of rigid body i is

$$g_i = g_{\mathrm{par}(i)} g_{\mathrm{par}(i),i}(q_i) \tag{2.7}$$

where $g_{\mathrm{par}(i),i}(q_i) = g_{\mathrm{par}(i),i}(0)\exp(\hat{S}_i q_i)$ and S_i is the body Jacobian of joint i with respect to frame $\{i\}$. In addition, the spatial Jacobian of joint i with respect to frame $\{0\}$ is

$$\overline{S}_i = \mathrm{Ad}_{g_i} S_i \tag{2.8}$$

in which S_i is constant by definition. Using Eqs. (2.7) and (2.8) as well as $\mathrm{Ad}_{g_i} S_i = \left(g_i \hat{S}_i g_i^{-1}\right)^\vee$, we obtain $\overline{\eta}_i = (\delta g_i g_i^{-1})^\vee$ as

$$\overline{\eta}_i = \overline{\eta}_{\mathrm{par}(i)} + \overline{S}_i \cdot \delta q_i, \tag{2.9}$$

or equivalently,

$$\overline{\eta}_i = \overline{S}_i \cdot \delta q_i + \sum_{j \in \mathrm{anc}(i)}^{n} \overline{S}_j \cdot \delta q_j \tag{2.10}$$

and furthermore,

$$\left(\frac{\partial g_i}{\partial q_j} g_i^{-1}\right)^\vee = \begin{cases} \overline{S}_j & j \in \mathrm{anc}(i) \cup \{i\}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.11a}$$

$$\left(\frac{\partial g_j}{\partial q_i} g_i^{-1}\right)^\vee = \begin{cases} \overline{S}_i & j \in \mathrm{des}(i) \cup \{i\}, \\ 0 & \text{otherwise}. \end{cases} \tag{2.11b}$$

In addition, from Eqs. (2.8) and (2.9), $\delta \mathrm{Ad}_{g_i} = \mathrm{ad}_{\overline{\eta}_i} \mathrm{Ad}_{g_i}$ and $\mathrm{ad}_{\overline{S}_i} \overline{S}_i = 0$, we obtain

$$\delta \overline{S}_i = \mathrm{ad}_{\overline{\eta}_i} \overline{S}_i = -\mathrm{ad}_{\overline{S}_i} \overline{\eta}_i = \mathrm{ad}_{\overline{\eta}_{\mathrm{par}(i)}} \overline{S}_i = -\mathrm{ad}_{\overline{S}_i} \overline{\eta}_{\mathrm{par}(i)}. \tag{2.12}$$

Moreover, as a result of Eqs. (2.10) to (2.12), we further obtain

$$\frac{\partial \overline{S}_i}{\partial q_j} = \begin{cases} \mathrm{ad}_{\overline{S}_j} \overline{S}_i & j \in \mathrm{anc}(i), \\ 0 & \text{otherwise}, \end{cases} \tag{2.13a}$$

$$\frac{\partial \overline{S}_j}{\partial q_i} = \begin{cases} \mathrm{ad}_{\overline{S}_i} \overline{S}_j & j \in \mathrm{des}(i), \\ 0 & \text{otherwise}. \end{cases} \tag{2.13b}$$

Since the spatial velocity $\overline{v}_i$ of rigid body i is

$$\overline{v}_i = \overline{S}_i \cdot \dot{q}_i + \sum_{j \in \mathrm{anc}(i)} \overline{S}_j \cdot \dot{q}_j = \overline{v}_{\mathrm{par}(i)} + \overline{S}_i \cdot \dot{q}_i, \tag{2.14}$$

we obtain

$$\begin{aligned} \delta \overline{v}_i &= \delta \overline{S}_i \cdot \dot{q}_i + \overline{S}_i \cdot \delta \dot{q}_i + \sum_{j \in \mathrm{anc}(i)} \left(\delta \overline{S}_j \cdot \dot{q}_j + \overline{S}_j \cdot \delta \dot{q}_j \right) \\ &= \delta \overline{v}_{\mathrm{par}(i)} + \delta \overline{S}_i \cdot \dot{q}_i + \overline{S}_i \cdot \delta \dot{q}_i. \end{aligned}$$

Substitute Eq. (2.12) into the equation above, the result is

$$\begin{aligned} \delta \overline{v}_i &= \mathrm{ad}_{\overline{\eta}_i} \overline{S}_i \cdot \dot{q}_i + \overline{S}_i \cdot \delta \dot{q}_i + \sum_{j \in \mathrm{anc}(i)} \left(\mathrm{ad}_{\overline{\eta}_j} \overline{S}_j \cdot \dot{q}_j + \overline{S}_j \cdot \delta \dot{q}_j \right) \\ &= \delta \overline{v}_{\mathrm{par}(i)} + \mathrm{ad}_{\overline{\eta}_i} \overline{S}_i \cdot \dot{q}_i + \overline{S}_i \cdot \delta \dot{q}_i. \end{aligned} \tag{2.15}$$

From Eqs. (2.12) to (2.15), we obtain

$$\frac{\partial \overline{v}_i}{\partial \dot{q}_j} = \begin{cases} S_j & j \in \mathrm{anc}(i) \cup \{i\}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.16a}$$

$$\frac{\partial \overline{v}_j}{\partial \dot{q}_i} = \begin{cases} S_i & j \in \mathrm{des}(i) \cup \{i\}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.16b}$$

and

$$\frac{\partial \overline{v}_i}{\partial q_j} = \begin{cases} \mathrm{ad}_{\overline{S}_j}(\overline{v}_i - \overline{v}_j) & j \in \mathrm{anc}(i) \cup \{i\}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.17a}$$

$$\frac{\partial \overline{v}_j}{\partial q_i} = \begin{cases} \mathrm{ad}_{\overline{S}_i}(\overline{v}_j - \overline{v}_i) & j \in \mathrm{des}(i) \cup \{i\}, \\ 0 & \text{otherwise}. \end{cases} \tag{2.17b}$$

In addition, from Eqs. (2.8) and (2.14), $\mathrm{Ad}_{\dot{g}_i} = \mathrm{ad}_{\overline{v}_i}\mathrm{Ad}_{g_i}$ and $\mathrm{ad}_{\overline{S}_i}\overline{S}_i = 0$, we obtain

$$\dot{\overline{S}}_i = \mathrm{ad}_{\overline{v}_i}\overline{S}_i = -\mathrm{ad}_{\overline{S}_i}\overline{v}_i = \mathrm{ad}_{\overline{v}_{\mathrm{par}(i)}}\overline{S}_i = -\mathrm{ad}_{\overline{S}_i}\overline{v}_{\mathrm{par}(i)}. \tag{2.18}$$

As for the spatial inertia matrix $\overline{M}_i = \mathrm{Ad}_{g_i}^{-T} M_i \mathrm{Ad}_{g_i}^{-1}$, algebraic manipulation shows that

$$\delta \overline{M}_i = -\mathrm{ad}_{\overline{\eta}_i}^T \cdot \overline{M}_i - \overline{M}_i \cdot \mathrm{ad}_{\overline{\eta}_i}, \tag{2.19}$$

and from Eqs. (2.9) to (2.11) and Eq. (2.19), we obtain

$$\frac{\partial \overline{M}_i}{\partial q_j} = \begin{cases} -\mathrm{ad}_{\overline{S}_j}^T \overline{M}_i - \overline{M}_i \mathrm{ad}_{\overline{S}_j} & j \in \mathrm{anc}(i) \cup \{i\}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.20a}$$

$$\frac{\partial \overline{M}_j}{\partial q_i} = \begin{cases} -\mathrm{ad}^T_{\overline{S}_i} \overline{M}_j - \overline{M}_j \mathrm{ad}_{\overline{S}_i} & j \in \mathrm{des}(i) \cup \{i\}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.20b}$$

In the rest of this chapter, Eq. (2.9) to (2.20) will be used to derive the algorithms and prove the propositions.

2.2.5. The Spatial Variation

In this subsection, we introduce the *spatial variation* $\overline{\delta}\,\overline{(\cdot)}$ that is used in the algorithms and the proof of the propositions. Note that the notion of the spatial variation $\overline{\delta}\,\overline{(\cdot)}$ only applies to the spatial quantities $\overline{(\cdot)}$ of $T_eSE(3)$ or $T^*_eSE(3)$ that are described in the spatial frame.

If $\overline{a}, a \in T_eSE(3)$ are related as $\overline{a} = \mathrm{Ad}_g a$ in which $g \in SE(3)$, we have

$$\delta \overline{a} = \mathrm{Ad}_g \delta a + \mathrm{ad}_{\overline{\eta}} \overline{a}$$

in which $\overline{\eta} = (\delta g g^{-1})^\vee$. For numerical simplicity, it is sometimes preferable to have the variations of $\overline{a}$ and a still related by Ad_g. Therefore, we define the spatial variation $\overline{\delta}\overline{a}$ to be

$$\overline{\delta}\overline{a} = \delta \overline{a} - \mathrm{ad}_{\overline{\eta}} \overline{a} \tag{2.21}$$

such that $\overline{\delta}\overline{a} = \mathrm{Ad}_g \delta a$ as long as $\overline{a} = \mathrm{Ad}_g a$. In a similar way, if $\overline{b}^*, b^* \in T^*_eSE(3)$ are related as $\overline{b}^* = \mathrm{Ad}_g^{-T} b^*$, we obtain

$$\delta \overline{b}^* = \mathrm{Ad}_g^{-T} \delta b^* - \mathrm{ad}^T_{\overline{\eta}} \overline{b}^*.$$

Similar to Eq. (2.21), the spatial variation $\overline{\delta}\,\overline{b}^*$ is defined to be

$$\overline{\delta}\,\overline{b}^* = \delta\overline{b}^* + \mathrm{ad}_{\overline{\eta}}^T\overline{b}^* \tag{2.22}$$

such that $\overline{\delta}\,\overline{b}^* = \mathrm{Ad}_g^{-T}\delta b^*$ as long as $\overline{b}^* = \mathrm{Ad}_g^{-T}b^*$. In addition, note that $\delta\big(b^{*T}a\big) = \delta b^{*T}a + b^{*T}\delta a = \overline{\delta}\,\overline{b}^{*T}\overline{a} + \overline{b}^{*T}\overline{\delta}\overline{a}$ and $\delta(\overline{b}^{*T}\overline{a}) = \delta(b^{*T}a)$, we have

$$\delta(\overline{b}^{*T}\overline{a}) = \overline{\delta}\,\overline{b}^{*T}\overline{a} + \overline{b}^{*T}\overline{\delta}\overline{a}. \tag{2.23}$$

In general, the spatial variations $\overline{\delta}\,\overline{(\cdot)}$ are the infinitesimal changes of spatial quantities in either the Lie algebra $T_eSE(3)$ or the dual Lie algebra $T_e^*SE(3)$ after canceling out the influences of the frame change.

In Section 2.3, we have a number of spatial quantities that are defined in $T_eSE(3)$ and $T_e^*SE(3)$, whose spatial variations $\overline{\delta}\,\overline{(\cdot)}$ can be computed in the tree representation.

Recall from Eqs. (2.8), (2.12) and (2.21) and $\overline{S}_i^{k,\alpha} = \mathrm{Ad}_{g_i^{k,\alpha}}S_i$ that the spatial variation $\overline{\delta}\,\overline{S}_i^{k,\alpha}$ is

$$\overline{\delta}\,\overline{S}_i^{k,\alpha} = 0 \tag{2.24}$$

though $\delta\overline{S}_i^{k,\alpha} = \mathrm{ad}_{\overline{\eta}_i^{k,\alpha}}\overline{S}_i^{k,\alpha}$ is usually not zero. In addition, according to Eqs. (2.15) and (2.21), we have

$$\overline{\delta}\overline{v}_i^{k,\alpha} = \delta\overline{v}_{\mathrm{par}(i)}^{k,\alpha} + \mathrm{ad}_{\overline{\eta}_i^{k,\alpha}}\overline{S}_i^{k,\alpha}\cdot\dot{q}_i^{k,\alpha} + \overline{S}_i^{k,\alpha}\cdot\delta\dot{q}_i^{k,\alpha} - \mathrm{ad}_{\overline{\eta}_i^{k,\alpha}}\overline{v}_i^{k,\alpha}$$

Substitute Eqs. (2.9) and (2.14) into the equation above to expand $\mathrm{ad}_{\overline{\eta}_i^{k,\alpha}}\overline{v}_i^{k,\alpha}$ and apply Eqs. (2.12) and (2.18), it can be shown that

$$\overline{\delta}\overline{v}_i^{k,\alpha} = \overline{\delta}\overline{v}_{\mathrm{par}(i)}^{k,\alpha} + \dot{\overline{S}}_i^{k,\alpha}\cdot\delta q_i^{k,\alpha} + \overline{S}_i^{k,\alpha}\cdot\delta\dot{q}^{k,\alpha}. \tag{2.25}$$

In terms of $\overline{\mu}_i^{k,\alpha}$, $H_i^{k,\alpha}$ and $\overline{\Omega}_i^{k,\alpha}$ in Eq. (2.31), which are spatial quantities in $T_e^*SE(3)$, we can still implement the tree representation to compute the spatial variation. According to Definition 2.2, we have

$$\delta\overline{\mu}_i^{k,\alpha} = \delta(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha}) + \sum_{j\in\mathrm{chd}(i)}\delta\overline{\mu}_j^{k,\alpha}.$$

From Eq. (2.22), the spatial variation $\overline{\delta}\overline{\mu}_i^{k,\alpha}$ is

$$\overline{\delta}\overline{\mu}_i^{k,\alpha} = \delta(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha}) + \sum_{j\in\mathrm{chd}(i)}\delta\overline{\mu}_j^{k,\alpha} + \mathrm{ad}^T_{\overline{\eta}_i^{k,\alpha}}\overline{\mu}_i^{k,\alpha}.$$

Using $\overline{\mu}_i^{k,\alpha} = \overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha} + \sum_{j\in\mathrm{chd}(i)}\overline{\mu}_j^{k,\alpha}$ and $\overline{\eta}_i^{k,\alpha} = \overline{\eta}_j^{k,\alpha} - \overline{S}_j^{k,\alpha}\cdot\delta q_j^{k,\alpha}$, we have

$$\begin{aligned}\overline{\delta}\overline{\mu}_i^{k,\alpha} = {} & \delta(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha}) + \mathrm{ad}^T_{\overline{\eta}_i^{k,\alpha}}(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha}) + \\ & \sum_{j\in\mathrm{chd}(i)}\left(\delta\overline{\mu}_j^{k,\alpha} + \mathrm{ad}^T_{\overline{\eta}_j^{k,\alpha}}\overline{\mu}_j^{k,\alpha} - \mathrm{ad}^T_{\overline{S}_j^{k,\alpha}}\overline{\mu}_j^{k,\alpha}\cdot\delta q_j^{k,\alpha}\right)\end{aligned} \tag{2.26}$$

As a result of Eqs. (2.19) and (2.21), $\delta(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha}) + \mathrm{ad}^T_{\overline{\eta}_i^{k,\alpha}}(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha})$ is

$$\delta(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha}) + \mathrm{ad}^T_{\overline{\eta}_i^{k,\alpha}}(\overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha}) = \overline{M}_i^{k,\alpha}(\delta\overline{v}_i^{k,\alpha} - \mathrm{ad}_{\overline{\eta}_i^{k,\alpha}}\overline{v}_i^{k,\alpha}) = \overline{M}_i^{k,\alpha}\overline{\delta}\overline{v}_i^{k,\alpha}. \tag{2.27}$$

From Eqs. (2.22) and (2.27) and $\mathrm{ad}^T_{\overline{S}_j^{k,\alpha}}\overline{\mu}_j^{k,\alpha} = \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}}\overline{S}_j^{k,\alpha}$, Eq. (2.26) is simplified to

$$
\begin{aligned}
\overline{\delta}\overline{\mu}_i^{k,\alpha} &= \overline{M}_i^{k,\alpha}\overline{\delta}\overline{v}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} \left(\overline{\delta}\overline{\mu}_j^{k,\alpha} - \mathrm{ad}^T_{\overline{S}_j^{k,\alpha}}\overline{\mu}_j^{k,\alpha}\cdot \delta q_j^{k,\alpha}\right) \\
&= \overline{M}_i^{k,\alpha}\overline{\delta}\overline{v}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} \left(\overline{\delta}\overline{\mu}_j^{k,\alpha} - \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}}\overline{S}_j^{k,\alpha}\cdot \delta q_j^{k,\alpha}\right).
\end{aligned} \tag{2.28}
$$

In a similar way, for the spatial variation $\overline{\delta}\, H_i^{k,\alpha}$, we obtain

$$
\begin{aligned}
\overline{\delta}\, H_i^{k,\alpha} &= \overline{\delta}\,\overline{F}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} \left(\overline{\delta}\, H_j^{k,\alpha} - \mathrm{ad}^T_{\overline{S}_j^{k,\alpha}} H_j^{k,\alpha}\cdot \delta q_j^{k,\alpha}\right) \\
&= \overline{\delta}\,\overline{F}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} \left(\overline{\delta}\, H_j^{k,\alpha} - \mathrm{ad}^D_{H_j^{k,\alpha}}\overline{S}_j^{k,\alpha}\cdot \delta q_j^{k,\alpha}\right).
\end{aligned} \tag{2.29}
$$

As for $\overline{\Omega}_i^{k,\alpha} = w^\alpha \Delta t\cdot \mathrm{ad}^T_{\overline{v}_i^{k,\alpha}}\cdot\overline{\mu}_i^{k,\alpha} + H_i^{k,\alpha}$, from Eqs. (2.21) and (2.22), algebraic manipulation shows that

$$
\begin{aligned}
\overline{\delta}\,\overline{\Omega}_i^{k,\alpha} &= \delta\overline{\Omega}_i^{k,\alpha} + \mathrm{ad}^T_{\overline{\eta}_i^{k,\alpha}}\overline{\Omega}_i^{k,\alpha} \\
&= w^\alpha\Delta t\cdot\left(\mathrm{ad}^T_{\overline{v}_i^{k,\alpha}}\cdot\overline{\delta}\overline{\mu}_i^{k,\alpha} + \mathrm{ad}^T_{\overline{\delta}\overline{v}_i^{k,\alpha}}\overline{\mu}_i^{k,\alpha}\right) + \overline{\delta}\, H_i^{k,\alpha} \\
&= w^\alpha\Delta t\cdot\left(\mathrm{ad}^T_{\overline{v}_i^{k,\alpha}}\cdot\overline{\delta}\overline{\mu}_i^{k,\alpha} + \mathrm{ad}^D_{\overline{\mu}_i^{k,\alpha}}\overline{\delta}\overline{v}_i^{k,\alpha}\right) + \overline{\delta}\, H_i^{k,\alpha}.
\end{aligned} \tag{2.30}
$$

We remark that Eqs. (2.24), (2.25) and (2.28) to (2.30) are fundamental to the algorithms and propositions in this chapter.

2.2.6. Differentiation on Lie Groups

For an analytical function $f : \mathbb{R}^n \to \mathbb{R}$, the directional derivative at $x \in \mathbb{R}^n$ in the direction δx is defined to be

$$
\mathbb{D}f(x)\cdot\delta x = \frac{\mathrm{d}}{\mathrm{d}t} f(x + t\cdot\delta x)\bigg|_{t=0}
$$

in which $\mathbb{D}f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T \in \mathbb{R}^n$.

In a similar way, we might define the directional derivative on Lie groups using the Lie algebra and the exponential map as follows.

Definition 2.1. If G is a n-dimensional smooth Lie group and $f : G \longrightarrow \mathbb{R}$ is a smooth function on G, the directional derivative at $g \in G$ in the direction $\overline{\eta} = \delta g g^{-1} \in T_e G$ is defined to be

$$\mathbb{D}f(g) \cdot \overline{\eta} = \frac{\mathrm{d}}{\mathrm{d}t} f\left(\exp\left(t \cdot \overline{\eta}\right) g\right)\Big|_{t=0}.$$

Moreover, if $\overline{e}_1, \overline{e}_2, \cdots, \overline{e}_n$ is a basis for the Lie algebra $T_e G$, then $\mathbb{D}f(g)$ can be explicitly written as

$$\mathbb{D}f(g) = \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} f\left(\exp\left(t \cdot \overline{e}_1\right) g\right) & f\left(\exp\left(t \cdot \overline{e}_2\right) g\right) & \cdots & f\left(\exp\left(t \cdot \overline{e}_n\right) g\right) \end{bmatrix}^T \Big|_{t=0}.$$

We remark that $\mathbb{R}^n$ is also a smooth Lie group for which the binary operation is addition, the Lie algebra is itself and the exponential map is the identity map. Furthermore, the definition of directional derivatives on Lie groups in Definition 2.1 is consistent with the definition of directional derivatives in $\mathbb{R}^n$. Therefore, it is without loss of any generality to interpret all the quantities in this chapter as elements of Lie groups and all the derivatives in this chapter as derivatives on Lie groups that are defined by Definition 2.1.

Following the notion in multivariate calculus, if $f : G_1 \times G_2 \times \cdots \times G_d \to \mathbb{R}$ is a smooth function in which G_1, G_2, $\cdots$, G_d are Lie groups, we use $\mathbb{D}_i f$ to denote the derivative with respect to G_i. In particular, for $\overline{F}_i^{k,\alpha} = \overline{F}_i^{k,\alpha}(g_i^{k,\alpha}, \overline{v}_i^{k,\alpha}, u_i^{k,\alpha})$ that is used to compute the Newton direction in Algorithm 3, note that $\mathbb{D}_1 \overline{F}_i^{k,\alpha}$ is the derivative with respect to $g_i^{k,\alpha}$ and $\mathbb{D}_2 \overline{F}_i^{k,\alpha}$ is the derivative with respect to $\overline{v}_i^{k,\alpha}$.

2.3. The Linear-Time Higher-Order Variational Integrator

In this and next sections, we present the propositions and algorithms efficiently computing higher-order variational integrators, whose derivations are omitted due to space limitations. Though not required for implementation, we refer the reader to Section 2.8 for detailed proofs.

In the rest of this chapter, if not specified, we assume that the mechanical system has n degrees of freedom and the higher-order variational integrator has $s+1$ control points $q^{k,\alpha} = q(t^{k,\alpha})$ in which $0 \leq \alpha \leq s$. Note that the notation $(\cdot)^{k,\alpha}$ is used to denote quantities $(\cdot)$ associated with $q^{k,\alpha}$ and $t^{k,\alpha}$, such as $q_i^{k,\alpha}$, $g_i^{k,\alpha}$, $\overline{v}_i^{k,\alpha}$, etc.

2.3.1. The DEL Equation Evaluation

To evaluate the DEL equations, the *discrete articulated body momentum* and *discrete articulated body impulse* are defined from the perspective of articulated bodies as follows.

Definition 2.2. The discrete articulated body momentum $\overline{\mu}_i^{k,\alpha} \in \mathbb{R}^6$ for articulated body i is defined to be $\overline{\mu}_i^{k,\alpha} = \overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha} + \sum_{j\in\mathrm{chd}(i)} \overline{\mu}_j^{k,\alpha}$ in which $\overline{M}_i^{k,\alpha}$ and $\overline{v}_i^{k,\alpha}$ are respectively the spatial inertia matrix and spatial velocity of rigid body i.

Definition 2.3. Suppose $\overline{F}_i(t) \in \mathbb{R}^6$ is the sum of all the wrenches directly acting on rigid body i, which does not include those applied or transmitted through the joints that are connected to rigid body i. The discrete articulated body impulse $H_i^{k,\alpha} \in \mathbb{R}^6$ for articulated body i is defined to be $H_i^{k,\alpha} = \overline{F}_i^{k,\alpha} + \sum_{j\in\mathrm{chd}(i)} H_j^{k,\alpha}$ in which $\overline{F}_i^{k,\alpha} = \omega^\alpha \overline{F}_i(t^{k,\alpha})\Delta t \in \mathbb{R}^6$ is the discrete impulse acting on rigid body i. Note that $\overline{F}_i(t)$, $\overline{F}_i^{k,\alpha}$ and $H_i^{k,\alpha}$ are expressed in frame $\{0\}$.

Remark 2.1. As for wrenches exerted on rigid body i, in addition to $\overline{F}_i(t)$ which includes gravity as well as the external wrenches that directly act on rigid body i, there are also wrenches applied through joints, e.g., from actuators, and wrenches transmitted through joints, e.g., from the parent and children of rigid body i in the tree representation.

It can be seen in Proposition 2.1 that $\overline{\mu}_i^{k,\alpha}$ and $H_i^{k,\alpha}$ make it possible to evaluate the DEL equations without explicitly calculating $\mathbb{D}_{\alpha+1}\mathcal{L}_d(\overline{q}^k)$ and $\mathcal{F}_d^{k,\alpha}$ in Eqs. (2.5) and (2.6).

Proposition 2.1. If $Q_i(t) \in \mathbb{R}$ is the sum of all joint forces applied to joint i and $p^k = \begin{bmatrix} p_1^k & p_2^k & \cdots & p_n^k \end{bmatrix}^T \in \mathbb{R}^n$ is the discrete momentum, the DEL equations Eq. (2.5) can be evaluated as

$$r_i^{k,0} = p_i^k + \overline{S}_i^{k,0^T} \cdot \overline{\Omega}_i^{k,0} + \sum_{\beta=0}^{s} a^{0\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,0}, \tag{2.31a}$$

$$r_i^{k,\alpha} = \overline{S}_i^{k,\alpha^T} \cdot \overline{\Omega}_i^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,\alpha} \quad \forall \alpha = 1, \cdots, s-1, \tag{2.31b}$$

$$p_i^{k+1} = \overline{S}_i^{k,s^T} \cdot \overline{\Omega}_i^{k,s} + \sum_{\beta=0}^{s} a^{s\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,s} \tag{2.31c}$$

in which $r_i^{k,\alpha}$ is the residue of the DEL equations Eqs. (2.5a) and (2.5b), $a^{\alpha\beta} = w^\beta b^{\beta\alpha}$, $\overline{\Omega}_i^{k,\alpha} = w^\alpha \Delta t \cdot \mathrm{ad}^T_{\overline{v}_i^{k,\alpha}} \cdot \overline{\mu}_i^{k,\alpha} + H_i^{k,\alpha}$, and $Q_i^{k,\alpha} = \omega^\alpha Q_i(t^{k,\alpha})\Delta t$ is the discrete joint force applied to joint i.

Proof. See Section 2.8.1. □

In Eqs. (2.31a) and (2.31b), if all $r_i^{k,\alpha}$ are equal to zero, a solution to the variational integrator as well as the DEL equations is obtained.

All the quantities used in Proposition 2.1 can be recursively computed in the tree representation, therefore, we have Algorithm 1 that evaluates the DEL equations, which essentially consists of $s+1$ forward passes from root to leaf nodes and $s+1$ backward passes in the reverse order, thus totally takes $O(sn)$ time. In contrast, automatic differentiation and our prior methods [**19**, **21**] take $O(sn^2)$ time to evaluate the DEL equations.

Algorithm 1 Recursive Evaluation of the DEL Equations

initialize $g_0^{k,\alpha} = \mathbf{I}$ and $\overline{v}_0^{k,\alpha} = 0$
for $i = 1 \to n$ **do**
  for $\alpha = 0 \to s$ **do**
    $g_i^{k,\alpha} = g_{\mathrm{par}(i)}^{k,\alpha} g_{\mathrm{par}(i),i}^{k,\alpha}(q_i^{k,\alpha})$
    $\overline{S}_i^{k,\alpha} = \mathrm{Ad}_{g_i^{k,\alpha}} S_i, \quad \overline{M}_i^{k,\alpha} = \mathrm{Ad}_{g_i^{k,\alpha}}^{-T} M_i \mathrm{Ad}_{g_i^{k,\alpha}}^{-1}$
    $\dot{q}_i^{k,\alpha} = \frac{1}{\Delta t} \sum_{\beta=0}^{s} b^{\alpha\beta} q_i^{k,\beta}, \quad \overline{v}_i^{k,\alpha} = \overline{v}_{\mathrm{par}(i)}^{k,\alpha} + \overline{S}_i^{k,\alpha} \cdot \dot{q}_i^{k,\alpha}$
  end for
end for
for $i = n \to 1$ **do**
  for $\alpha = 0 \to s$ **do**
    $\overline{\mu}_i^{k,\alpha} = \overline{M}_i^{k,\alpha} \overline{v}_i^{k,\alpha} + \sum_{j \in \mathrm{chd}(i)} \overline{\mu}_j^{k,\alpha}, \quad H_i^{k,\alpha} = \overline{F}_i^{k,\alpha} + \sum_{j \in \mathrm{chd}(i)} H_j^{k,\alpha}$
    $\overline{\Omega}_i^{k,\alpha} = w^\alpha \Delta t \cdot \mathrm{ad}_{\overline{v}_i^{k,\alpha}}^T \cdot \overline{\mu}_i^{k,\alpha} + H_i^{k,\alpha}$
  end for
  $r_i^{k,0} = p_i^k + \overline{S}_i^{k,0^T} \overline{\Omega}_i^{k,0} + \sum_{\beta=0}^{s} a^{0\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,0}$
  for $\alpha = 1 \to s-1$ **do**
    $r_i^{k,\alpha} = \overline{S}_i^{k,\alpha^T} \overline{\Omega}_i^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,\alpha}$
  end for
  $p_i^{k+1} = \overline{S}_i^{k,s^T} \overline{\Omega}_i^{k,s} + \sum_{\beta=0}^{s} a^{s\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,s}$
end for

2.3.2. Exact Newton Direction Computation

From Eq. (2.5), the Newton direction $\delta\overline{q}^k = \begin{bmatrix}\delta q^{k,1^T}, \cdots, \delta q^{k,s^T}\end{bmatrix}^T \in \mathbb{R}^{sn}$ is computed as $\delta\overline{q}^k = -\mathcal{J}^{k^{-1}}(\overline{q}^k)\cdot r^k$ in which $\mathcal{J}^k(\overline{q}^k) \in \mathbb{R}^{sn\times sn}$ is the Jacobian of Eqs. (2.5a) and (2.5b) with respect to control points $q^{k,1}, \cdots, q^{k,s}$, and $r^k \in \mathbb{R}^{sn}$ is the residue of evaluating the DEL equations Eqs. (2.5a) and (2.5b) by Proposition 2.1.

In this chapter, we make the the following assumption on $\overline{F}_i^{k,\alpha}$ and $Q_i^{k,\alpha}$, which is general and applies to a large number of mechanical systems in robotics.

Assumption 2.1. Let $u(t)$ be the control inputs of the mechanical system, we assume that the discrete impulse $\overline{F}_i^{k,\alpha}$ and discrete joint force $Q_i^{k,\alpha}$ can be respectively formulated as $\overline{F}_i^{k,\alpha} = \overline{F}_i^{k,\alpha}(g_i^{k,\alpha}, \overline{v}_i^{k,\alpha}, u^{k,\alpha})$ and $Q_i^{k,\alpha} = Q_i^{k,\alpha}(q_i^{k,\alpha}, \dot{q}_i^{k,\alpha}, u^{k,\alpha})$ in which $u^{k,\alpha} = u(t^{k,\alpha})$.

If Assumption 2.1 holds and $\mathcal{J}^{k^{-1}}(\overline{q}^k)$ exists, it can be shown that Algorithm 2 computes the Newton direction for variational integrators in $O(s^3n)$ time.

Proposition 2.2. For higher-order variational integrators of unconstrained mechanical systems, if Assumption 2.1 holds and $\mathcal{J}^{k^{-1}}(\overline{q}^k)$ exists, the Newton direction $\delta\overline{q}^k = -\mathcal{J}^{k^{-1}}(\overline{q}^k)\cdot r^k$ can be computed with Algorithm 2 in $O(s^3n)$ time.

Proof. See Section 2.8.2. □

In Algorithm 2, the forward and backward passes of the tree structure take $O(s^2n)$ time, and the n computations of the $s \times s$ matrix inverse takes $O(s^3n)$ time, thus the overall complexity of Algorithm 2 is $O(s^3n + s^2n)$. In contrast, automatic differentiation and our prior methods in [**19**, **21**] take $O(s^2n^3)$ time to compute $\mathcal{J}^k(\overline{q}^k)$ and another $O(s^3n^3)$ time to compute the $sn \times sn$ matrix inverse $\mathcal{J}^{k^{-1}}(\overline{q}^k)$, and the overall complexity

Algorithm 2 Recursive Computation of the Newton Direction

initialize $g_0^{k,\alpha} = \mathbf{I}$ and $\overline{v}_0^{k,\alpha} = 0$
for $i = 1 \to n$ **do**
  for $\alpha = 0 \to s$ **do**
    $g_i^{k,\alpha} = g_{\mathrm{par}(i)}^{k,\alpha} g_{\mathrm{par}(i),i}^{k,\alpha}(q_i^{k,\alpha})$
    $\overline{S}_i^{k,\alpha} = \mathrm{Ad}_{g_i^{k,\alpha}} S_i, \quad \overline{M}_i^{k,\alpha} = \mathrm{Ad}_{g_i^{k,\alpha}}^{-T} M_i \mathrm{Ad}_{g_i^{k,\alpha}}^{-1}$
    $\dot{q}_i^{k,\alpha} = \dfrac{1}{\Delta t} \sum_{\beta=0}^{s} b^{\alpha\beta} q_i^{k,\beta}, \quad \overline{v}_i^{k,\alpha} = \overline{v}_{\mathrm{par}(i)}^{k,\alpha} + \overline{S}_i^{k,\alpha} \cdot \dot{q}_i^{k,\alpha}$
    $\dot{\overline{S}}_i^{k,\alpha} = \mathrm{ad}_{\overline{v}_i^{k,\alpha}} \overline{S}_i^{k,\alpha}$
  end for
end for
for $i = n \to 1$ **do**
  use Algorithm 3 to evaluate
    a) $D_i^{k,\alpha\rho}$, $G_i^{k,\alpha\nu}$, $l_i^{k,\alpha}$ and $\overline{\mu}_i^{k,\alpha}$
    b) $\Pi_i^{k,\alpha\rho}$, $\Psi_i^{k,\alpha\nu}$, $\zeta_i^{k,\alpha}$ and $H_i^{k,\alpha}$
    c) $H_i^{k,\alpha}$ and $\Phi_i^{k,\alpha}$
    d) $X_i^{k,\alpha\rho}$, $Y_i^{k,\alpha\nu}$ and $y_i^{k,\alpha}$
end for
initialize $\overline{\eta}_0^{k,\nu} = 0$ and $\overline{\delta}\overline{v}_0^{k,\rho} = 0$
for $i = 1 \to n$ **do**
  for $\gamma = 1 \to s$ **do**
    $\delta q_i^{k,\gamma} = \sum_{\rho=0}^{s} X_i^{k,\gamma\rho} \cdot \overline{\delta}\overline{v}_{\mathrm{par}(i)}^{k,\rho} + \sum_{\nu=1}^{s} Y_i^{k,\gamma\nu} \cdot \overline{\eta}_{\mathrm{par}(i)}^{k,\nu} + y_i^{k,\gamma}$
  end for
  for $\nu = 1 \to s$ **do**
    $\overline{\eta}_i^{k,\nu} = \overline{\eta}_{\mathrm{par}(i)}^{k,\nu} + \overline{S}_i^{k,\nu} \cdot \delta q_i^{k,\nu}$
  end for
  for $\rho = 0 \to s$ **do**
    $\delta \dot{q}_i^{k,\rho} = \dfrac{1}{\Delta t} \sum_{\gamma=1}^{s} b^{\rho\gamma} \cdot \delta q_i^{k,\gamma}$
    $\overline{\delta}\overline{v}_i^{k,\rho} = \overline{\delta}\overline{v}_{\mathrm{par}(i)}^{k,\rho} + \dot{\overline{S}}_i^{k,\rho} \cdot \delta q_i^{k,\rho} + \overline{S}_i^{k,\rho} \cdot \delta \dot{q}_i^{k,\rho}$
  end for
end for

Algorithm 3 Recursive Computation of the Newton Direction – Backward Pass

1: $\forall \alpha = 0,\, 1,\, \cdots,\, s$, $\forall \rho = 0,\, 1,\, \cdots,\, s$ and $\forall \nu = 0,\, 1,\, \cdots,\, s-1$,

$$D_i^{k,\alpha\rho} = \sigma^{\alpha\rho}\overline{M}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} \Big(D_j^{k,\alpha\rho} + \sum_{\gamma=1}^{s} H_j^{k,\alpha\gamma} X_j^{k,\gamma\rho} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} X_j^{k,\alpha\rho} \Big), \tag{2.32a}$$

$$G_i^{k,\alpha\nu} = \sum_{j\in \mathrm{chd}(i)} \Big(G_j^{k,\alpha\nu} + \sum_{\gamma=1}^{s} H_j^{k,\alpha\gamma} Y_j^{k,\gamma\nu} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} Y_j^{k,\alpha\nu} \Big), \tag{2.32b}$$

$$l_i^{k,\alpha} = \sum_{j\in \mathrm{chd}(i)} \Big(l_j^{k,\alpha} + \sum_{\gamma=1}^{s} H_j^{k,\alpha\gamma} y_j^{k,\gamma} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} y_j^{k,\alpha} \Big), \tag{2.32c}$$

$$\overline{\mu}_i^{k,\alpha} = \overline{M}_i^{k,\alpha} \overline{v}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} \mu_j^{k,\alpha} \tag{2.32d}$$

where

$$\sigma^{\alpha\rho} = \begin{cases} 1 & \alpha = \rho, \\ 0 & \alpha \neq \rho \end{cases} \qquad \text{and} \qquad \overline{\sigma}^{\alpha 0} = \begin{cases} 1 & \alpha \neq 0, \\ 0 & \alpha = 0 \end{cases} \tag{2.33}$$

2: $\forall \alpha = 0,\, 1,\, \cdots,\, s-1$, $\forall \rho = 0,\, 1,\, \cdots,\, s$ and $\forall \nu = 0,\, 1,\, \cdots,\, s-1$,

$$\Pi_i^{k,\alpha\rho} = \sigma^{\alpha\rho} \mathbb{D}_2 \overline{F}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} \Big(\Pi_j^{k,\alpha\rho} + \sum_{\gamma=1}^{s} \Phi_j^{k,\alpha\gamma} X_j^{k,\gamma\rho} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} X_j^{k,\alpha\rho} \Big), \tag{2.34a}$$

$$\begin{aligned} \Psi_i^{k,\alpha\nu} = {} & \sigma^{\alpha\nu} \Big(\mathbb{D}_1 \overline{F}_i^{k,\alpha} + \mathrm{ad}^D_{\overline{F}_i^{k,\alpha}} - \mathbb{D}_2 \overline{F}_i^{k,\alpha} \mathrm{ad}_{\overline{v}_i^{k,\alpha}} \Big) + \\ & \sum_{j\in \mathrm{chd}(i)} \Big(\Psi_j^{k,\alpha\nu} + \sum_{\gamma=1}^{s} \Phi_j^{k,\alpha\gamma} Y_j^{k,\gamma\nu} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} Y_j^{k,\alpha\nu} \Big), \end{aligned} \tag{2.34b}$$

$$\zeta_i^{k,\alpha} = \sum_{j\in \mathrm{chd}(i)} \Big(\zeta_j^{k,\alpha} + \sum_{\gamma=1}^{s} \Phi_j^{k,\alpha\gamma} y_j^{k,\gamma} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} y_j^{k,\alpha} \Big), \tag{2.34c}$$

$$H_i^{k,\alpha} = \overline{F}_i^{k,\alpha} + \sum_{j\in \mathrm{chd}(i)} H_j^{k,\alpha} \tag{2.34d}$$

3: $\forall \alpha = 0,\, 1,\, \cdots,\, s$ and $\forall \gamma = 1,\, 2,\, \cdots,\, s$,

$$H_i^{k,\alpha\gamma} = D_i^{k,\alpha\gamma} \dot{\overline{S}}_i^{k,\gamma} + G_i^{k,\alpha\gamma} \overline{S}_i^{k,\gamma} + \frac{1}{\Delta t} \sum_{\rho=0}^{s} b^{\rho\gamma} D_i^{k,\alpha\rho} \overline{S}_i^{k,\rho}.$$

4: $\forall \alpha = 0,\, 1,\, \cdots,\, s-1$ and $\forall \gamma = 1,\, 2,\, \cdots,\, s$,

$$\Phi_i^{k,\alpha\gamma} = \Pi_i^{k,\alpha\gamma} \dot{\overline{S}}_i^{k,\gamma} + \Psi_i^{k,\alpha\gamma} \overline{S}_i^{k,\gamma} + \frac{1}{\Delta t} \sum_{\rho=0}^{s} b^{\rho\gamma} \Pi_i^{k,\alpha\rho} \overline{S}_i^{k,\rho}.$$

5: $\forall \alpha = 0,\, 1,\, \cdots,\, s-1,\, \forall \rho = 0,\, 1,\, \cdots,\, s$ and $\forall \nu = 0,\, 1,\, \cdots,\, s-1,$

$$\Theta_i^{k,\alpha\rho} = w^\alpha \Delta t \cdot \left(\dot{\overline{S}}_i^{k,\alpha^T} D_i^{k,\alpha\rho} + \sigma^{\alpha\rho} \overline{S}_i^{k,\alpha^T} \mathrm{ad}^D_{\mu_i^{k,\alpha}}\right) + \overline{S}_i^{k,\alpha^T} \Pi_i^{k,\alpha\rho},$$

$$\Xi_i^{k,\alpha\nu} = w^\alpha \Delta t \cdot \dot{\overline{S}}_i^{k,\alpha^T} G_i^{k,\alpha\nu} + \overline{S}_i^{k,\alpha^T} \Psi_i^{k,\alpha\nu}.$$

6: $\forall \alpha = 0,\, 1,\, \cdots,\, s-1,\, \forall \rho = 0,\, 1,\, \cdots,\, s$ and $\forall \nu = 0,\, 1,\, \cdots,\, s-1,$

$$\overline{\Theta}_i^{k,\alpha\rho} = \Theta_i^{k,\alpha\rho} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} D_i^{k,\beta\rho},$$

$$\overline{\Xi}_i^{k,\alpha\nu} = \Xi_i^{k,\alpha\nu} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} G_i^{k,\beta\nu},$$

$$\overline{\xi}_i^{k,\alpha} = w^\alpha \Delta t \cdot \dot{\overline{S}}_i^{k,\alpha^T} l_i^{k,\alpha} + \overline{S}_i^{k,\alpha^T} \zeta_i^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} l_i^{k,\beta}.$$

7: $\forall \alpha = 0,\, 1,\, \cdots,\, s-1$ and $\forall \gamma = 1,\, 2,\, \cdots,\, s,$

$$\Lambda_i^{k,\alpha\gamma} = w^\alpha \Delta t \cdot \dot{\overline{S}}_i^{k,\alpha^T} H_i^{k,\alpha\gamma} + \overline{S}_i^{k,\alpha^T} \Phi_i^{k,\alpha\gamma} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} H_i^{k,\beta\gamma} + \sigma^{\alpha\gamma} \left(\mathbb{D}_1 Q_i^{k,\alpha} + w^\alpha \Delta t \cdot \overline{S}_i^{k,\alpha^T} \mathrm{ad}^D_{\mu_i^{k,\alpha}} \dot{\overline{S}}_i^{k,\alpha} \right) + \frac{1}{\Delta t} b^{\alpha\gamma} \cdot \mathbb{D}_2 Q_i^{k,\alpha}$$

with which $\Lambda_i^k = \left[\Lambda_i^{k,\alpha\gamma}\right] \in \mathbb{R}^{s\times s}$

8: $\forall \gamma = 1,\, 2,\, \cdots,\, s$ and $\forall \varrho = 0,\, 1,\, \cdots,\, s-1$, compute $\overline{\Lambda}_i^{k,\gamma\varrho}$ such that $\Lambda_i^{k^{-1}} = \left[\overline{\Lambda}_i^{k,\gamma\varrho}\right] \in \mathbb{R}^{s\times s}$

9: $\forall \gamma = 1,\, 2,\, \cdots,\, s,\, \forall \rho = 0,\, 1,\, \cdots,\, s$ and $\forall \nu = 1,\, 2,\, \cdots,\, s$

$$X_i^{k,\gamma\rho} = -\sum_{\varrho=0}^{s-1} \overline{\Lambda}_i^{k,\gamma\varrho} \cdot \overline{\Theta}_i^{k,\varrho\rho},$$

$$Y_i^{k,\gamma\nu} = -\sum_{\varrho=0}^{s-1} \overline{\Lambda}_i^{k,\gamma\varrho} \cdot \overline{\Xi}_i^{k,\varrho\nu},$$

$$y_i^{k,\gamma} = -\sum_{\varrho=0}^{s-1} \overline{\Lambda}_i^{k,\gamma\varrho} \left(r_i^{k,\varrho} + \overline{\xi}_i^{k,\varrho} \right)$$

is $O(s^3n^3 + s^2n^3)$. Though the quasi-Newton method [**4**] is $O(n)$ time for second-order variational integrator in which $s = 1$, it requires small time steps and can not be used for third- or higher-order variational integrators.

Therefore, both Algorithms 1 and 2 have $O(n)$ complexity for a given s, which results in a linear-time variational integrator. Furthermore, Algorithms 1 and 2 have no restrictions on the number of control points, which indicates that the resulting linear-time variational integrator can be arbitrarily high order. *To our knowledge, this is the first exactly linear-time third- or higher-order implicit integrator for mechanical systems.*

2.3.3. Extension to Constrained Mechanical Systems

Thus far all our discussions of linear-time variational integrators have been restricted to unconstrained mechanical systems. However, Algorithms 1 and 2 can be extended to constrained mechanical systems as well.

In terms of the the DEL equation evaluation, the extension to constrained mechanical systems is immediate. From Eq. (2.6), we only need to add the constraint term $A^{k,\alpha}(q^{k,\alpha}) \cdot \lambda^{k,\alpha}$ to the results of using Algorithm 1.

If the variational integrator is second-order and the mechanical system has m constraints, it is possible to compute the Newton direction δq^{k+1} and $\delta\lambda^k$ in $O(mn) + O(m^3)$ time using Algorithm 2. In accordance with Eq. (2.6), δq^{k+1} and $\delta\lambda^k$ should satisfy $\mathcal{J}^k(q^k) \cdot \delta q^{k+1} + A^k(q^k) \cdot \delta\lambda^k = -r_q^k$ and $\mathbb{D}h^k(q^{k+1}, \dot{q}^{k+1}) \cdot \delta q^{k+1} = -r_c^k$ in which r_q^k and r_c^k are equation residues. Then δq^{k+1} and $\delta\lambda^k$ can be computed as follows: i) compute $\delta q_r^{k+1} = -\mathcal{J}^{k^{-1}} \cdot r_q^k$ with Algorithm 2 which takes $O(n)$ time; ii) compute $\mathcal{J}^{k^{-1}} \cdot A^k$ by using Algorithm 2 m times which takes $O(mn)$ time; iii) compute $\delta\lambda^k = \left(\mathbb{D}h^k \cdot \mathcal{J}^k \cdot A^k\right)^{-1}(r_c^k + \mathbb{D}h^k \cdot \delta q_r^{k+1})$ which takes $O(m^3)$ time; iv) compute $\delta q^{k+1} = \delta q_r^{k+1} - \mathcal{J}^{k^{-1}} \cdot A^k \cdot \delta\lambda^k$.

In regard to third- or higher-order variational integrators, if the constraints are of $h_i^k(g_i^{k,\alpha}, \overline{v}_i^{k,\alpha}) = 0$ or $h_i^k(q_i^{k,\alpha}, \dot{q}_i^{k,\alpha}) = 0$ or both for each $i = 1, 2, \cdots, n$, Algorithm 2 can be used to compute the Newton direction $\delta\overline{q}^k$ and $\delta\lambda^k$ in a similar procedure to the second-order variational integrator.

In next section, we will discuss the linearization of higher-order variational integrators in $O(n^2)$ time.

2.4. The Linearization of Higher-Order Variational Integrators

The linearization of discrete time systems is useful for trajectory optimization, stability analysis, controller design, etc., which are import tools in robotics.

From Eqs. (2.5) and (2.6), the linearization of variational integrators is comprised of the computation of $\mathbb{D}^2\mathcal{L}_d(\overline{q}^k)$, $\mathbb{D}\mathcal{F}_d^{k,\alpha}(t^{k,\alpha})$ and $\mathbb{D}A^{k,\alpha}(q^{k,\alpha})$. In most cases, $\mathbb{D}\mathcal{F}_d^{k,\alpha}(t^{k,\alpha})$ and $\mathbb{D}A^{k,\alpha}(q^{k,\alpha})$ can be efficiently computed in $O(n^2)$ time, therefore, the linearization efficiency is mostly affected by $\mathbb{D}^2\mathcal{L}_d(\overline{q}^k)$.

It is by definition that the Lagrangian of a mechanical system is $\mathcal{L}(q, \dot{q}) = K(q, \dot{q}) - V(q)$ in which $K(q, \dot{q})$ is the kinetic energy and $V(q)$ is the potential energy, and from Eq. (2.4), the computation of $\mathbb{D}^2\mathcal{L}_d(\overline{q}^k)$ is actually to compute $\frac{\partial K}{\partial \dot{q}^2}$, $\frac{\partial^2 K}{\partial \dot{q} \partial q}$, $\frac{\partial^2 K}{\partial q \partial \dot{q}}$, $\frac{\partial^2 K}{\partial q^2}$ and $\frac{\partial V}{\partial q^2}$, for which we have Propositions 2.3 and 2.4 as follows.

Proposition 2.3. For the kinetic energy $K(q, \dot{q})$ of a mechanical system, $\frac{\partial^2 K}{\partial \dot{q}^2}$, $\frac{\partial^2 K}{\partial \dot{q} \partial q}$, $\frac{\partial^2 K}{\partial q \partial \dot{q}}$, $\frac{\partial^2 K}{\partial q^2}$ can be recursively computed with Algorithm 4 in $O(n^2)$ time.

Proof. See Section 2.8.3. □

In the matter of potential energy $V(q)$, we only consider the gravitational potential energy $V_{\vec{g}}(q)$, and the other types of potential energy can be computed in a similar way.

Algorithm 4 Recursive Computation of $\frac{\partial^2 K}{\partial \dot{q}^2}$, $\frac{\partial^2 K}{\partial \dot{q} \partial q}$, $\frac{\partial^2 K}{\partial q \partial \dot{q}}$, $\frac{\partial^2 K}{\partial q^2}$

initialize $g_0 = \mathbf{I}$ and $\overline{v}_0 = 0$
for $i = 1 \to n$ **do**
  $g_i = g_{\mathrm{par}(i)} g_{\mathrm{par}(i),i}(q_i)$
  $\overline{M}_i = \mathrm{Ad}_{g_i}^{-T} M_i \mathrm{Ad}_{g_i}^{-1}, \quad \overline{S}_i = \mathrm{Ad}_{g_i} S_i$
  $\overline{v}_i = \overline{v}_{\mathrm{par}(i)} + \overline{S}_i \cdot \dot{q}_i, \quad \dot{\overline{S}}_i = \mathrm{ad}_{\overline{v}_i} \overline{S}_i$
end for
initialize $\frac{\partial^2 K}{\partial \dot{q}^2} = \mathbf{0}, \quad \frac{\partial^2 K}{\partial \dot{q} \partial q} = \mathbf{0}, \quad \frac{\partial^2 K}{\partial q \partial \dot{q}} = \mathbf{0}, \quad \frac{\partial^2 K}{\partial q^2} = \mathbf{0}$
for $i = n \to 1$ **do**
  $\overline{\mu}_i = \overline{M}_i \overline{v}_i + \sum_{j \in \mathrm{chd}(i)} \overline{\mu}_j, \quad \overline{\mathcal{M}}_i = \overline{M}_i + \sum_{j \in \mathrm{chd}(i)} \overline{\mathcal{M}}_j$
  $\overline{\mathcal{M}}_i^A = \overline{\mathcal{M}}_i \overline{S}_i, \quad \overline{\mathcal{M}}_i^B = \overline{\mathcal{M}}_i \dot{\overline{S}}_i - \mathrm{ad}_{\overline{\mu}_i}^D \overline{S}_i$
  for $j \in \mathrm{anc}(i) \cup \{i\}$ **do**
    $\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j} = \frac{\partial^2 K}{\partial \dot{q}_j \partial \dot{q}_i} = \overline{S}_j^T \overline{\mathcal{M}}_i^A$
    $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j} = \frac{\partial^2 K}{\partial q_j \partial \dot{q}_i} = \dot{\overline{S}}_j^T \overline{\mathcal{M}}_i^A, \quad \frac{\partial^2 K}{\partial q_i \partial \dot{q}_j} = \frac{\partial^2 K}{\partial \dot{q}_j \partial q_i} = \overline{S}_j^T \overline{\mathcal{M}}_i^B$
    $\frac{\partial^2 K}{\partial q_i \partial q_j} = \frac{\partial^2 K}{\partial q_j \partial q_i} = \dot{\overline{S}}_j^T \overline{\mathcal{M}}_i^B$
  end for
end for

Proposition 2.4. If $\vec{g} \in \mathbb{R}^3$ is gravity, then for the gravitational potential energy $V_{\vec{g}}(q)$, $\frac{\partial^2 V_{\vec{g}}}{\partial q^2}$ can be recursively computed with Algorithm 5 in $O(n^2)$ time.

Proof. See Section 2.8.4. □

In regard to Proposition 2.4 and Algorithm 5, we remind the reader of the notation introduced in Sections 2.2.2 and 2.2.3 that $m_i \in \mathbb{R}$ is the mass of rigid body i, $p_i \in \mathbb{R}^3$ is the mass center of rigid body i as well as the origin of frame $\{i\}$, and $\overline{S}_i = \begin{bmatrix} \overline{s}_i^T & \overline{n}_i^T \end{bmatrix}^T \in \mathbb{R}^6$ is the spatial Jacobian of joint i with respect to frame $\{0\}$.

If $\frac{\partial K}{\partial \dot{q}^2}$, $\frac{\partial^2 K}{\partial \dot{q} \partial q}$, $\frac{\partial^2 K}{\partial q \partial \dot{q}}$, $\frac{\partial^2 K}{\partial q^2}$ and $\frac{\partial V}{\partial q^2}$ are computed in $O(n^2)$ time, then according to Eqs. (2.2) and (2.4), the remaining computation of $\mathbb{D}^2 \mathcal{L}_d(\overline{q}^{k,\alpha})$ is simply the application of the chain rule. Therefore, if the variational integrator has $s+1$ control points, the complexity of

Algorithm 5 Recursive Computation of $\frac{\partial^2 V_{\vec{g}}}{\partial q^2}$

initialize $g_0 = \mathbf{I}$
for $i = 1 \to n$ **do**
    $g_i = g_{\mathrm{par}(i)} g_{\mathrm{par}(i),i}(q_i), \quad \overline{S}_i = \mathrm{Ad}_{g_i} S_i$
end for
initialize $\frac{\partial^2 V_{\vec{g}}}{\partial q^2} = \mathbf{0}$
for $i = n \to 1$ **do**
    $\overline{\sigma}_{m_i} = m_i + \sum_{j \in \mathrm{chd}(i)} \overline{\sigma}_{m_j}, \quad \overline{\sigma}_{p_i} = m_i p_i + \sum_{j \in \mathrm{chd}(i)} \overline{\sigma}_{p_j}$
    $\overline{\sigma}_i^A = \hat{\vec{g}} \left(\overline{\sigma}_{m_i} \cdot \overline{n}_i - \hat{\overline{\sigma}}_{p_i} \cdot \overline{s}_i \right)$
    for $j \in \mathrm{anc}(i) \cup \{i\}$ **do**
        $\frac{\partial^2 V_{\vec{g}}}{\partial q_i \partial q_j} = \frac{\partial^2 V_{\vec{g}}}{\partial q_j \partial q_i} = \overline{s}_j^T \cdot \overline{\sigma}_i^A$
    end for
end for

the linearization is $O(s^2 n^2)$. In contrast, automatic differentiation and our prior methods [**19**, **21**] take $O(s^2 n^3)$ time to linearize the variational integrators.

2.5. Comparison with Existing Methods

The variational integrators using Algorithms 1 to 5 are compared with the linear-time quasi-Newton method [**4**], automatic differentiation and the Hermite-Simpson direct collocation method, which verifies the accuracy, efficiency and scalability of our work. All the tests are run in C++ on a 3.1GHz Intel Core Xeon Thinkpad P51 laptop.

2.5.1. Comparison with the Linear-Time Quasi-Newton Method

In this subsection, we compare the $O(n)$ Newton method using Algorithms 1 and 2 with the $O(n)$ quasi-Newton method in [**4**] on the trapezoidal variational integrator (Example 2.1) of a 32-link pendulum with different time steps.

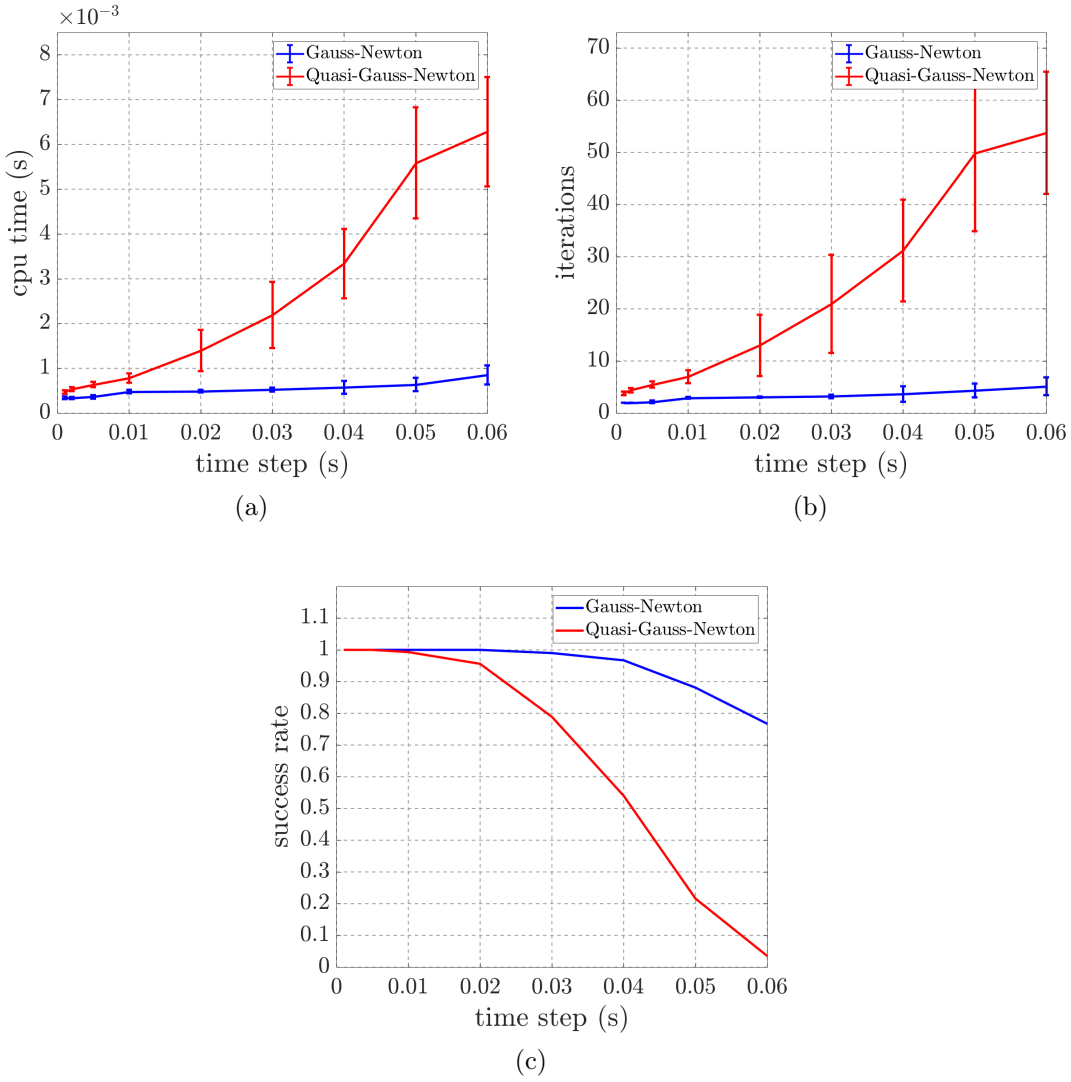


Figure 2.1. The comparison of the $O(n)$ Newton method with the $O(n)$ quasi-Newton method [**4**] for the trapezoidal variational integrator of a 32-link pendulum with different time steps. The results of computational time are in (a), number of iterations in (b) and success rates in (c). Each result is calculated over 1000 initial conditions.

In the comparison, 1000 initial joint angles q^0 and joint velocities $\dot{q}^0$ are uniformly sampled from $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for each of the selected time steps, which are 0.001s, 0.002s, 0.005s, 0.01s, 0.02s, 0.03s, 0.04s, 0.05s and 0.06s, and the Newton and quasi-Newton methods

are used to solve the DEL equations for one time step. The results are in Fig. 2.1, in which the computational time and the number of iterations are calculated only over initial conditions that the DEL equations are successfully solved. It can be seen that the Newton method using Algorithms 1 and 2 outperforms the quasi-Newton method in [**4**] in all aspects, especially for relatively large time steps.

2.5.2. Comparison with Automatic Differentiation

In this subsection, we compare Algorithms 1 to 5 with automatic differentiation for evaluating the DEL equations, computing the Newton direction and linearizing the DEL equations. The variational integrator used is the Simpson variational integrator (Example 2.2).

In the comparison, we use pendulums with different numbers of links as benchmark systems. For each pendulum, 100 initial joint angles q^0 and joint velocities $\dot{q}^0$ are uniformly sampled from $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The results are in Fig. 2.2 and it can be seen that our recursive algorithms are much more efficient, which is consistent with the fact that Algorithms 1 to 5 are $O(n)$ for evaluating the DEL equations, $O(n)$ for computing the Newton direction, and $O(n^2)$ for linearizing the DEL equations, whereas automatic differentiation are $O(n^2)$, $O(n^3)$ and $O(n^3)$, respectively.

2.5.3. Comparison with the Hermite-Simpson Direct Collocation Method

In this subsection, we compare the fourth-order Simpson variational integrator (Example 2.2) with the Hermite-Simpson direct collocation method, which is a third-order

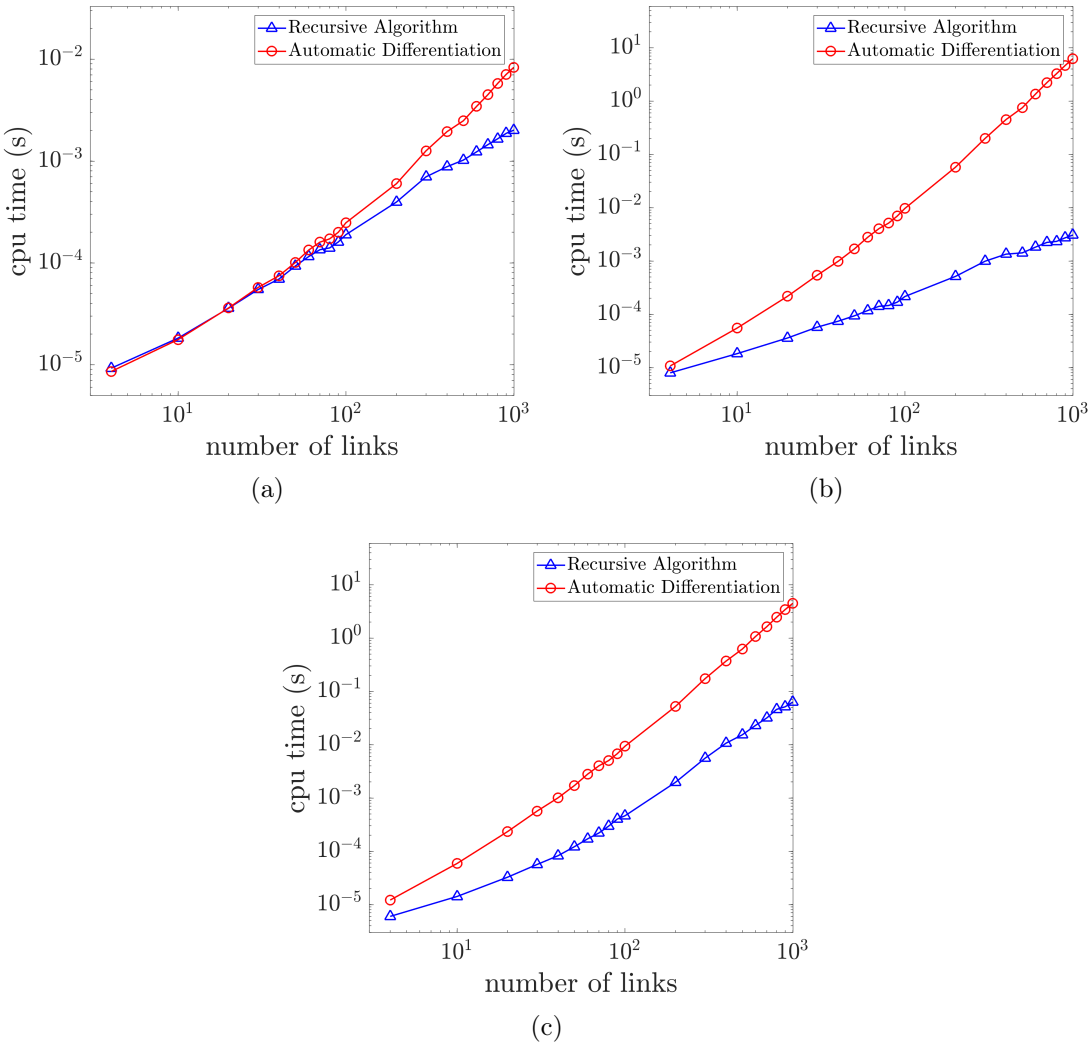


Figure 2.2. The comparison of our recursive algorithms with automatic differentiation for pendulums with different numbers of links. The variational integrator used is the Simpson variational integrator. The results of evaluating the DEL equations are in (a), computing the Newton direction in (b) and linearizing the DEL equations in (c). Each result is calculated over 100 initial conditions.

implicit integrator commonly used in robotics for trajectory optimization [**26**, **27**].[1] Note that both integrators use three control points for integration.

[1]The Hermite-Simpson direct collocation methods used in [**26**, **27**] are actually implicit integrators that integrate the trajectory as a second-order system in the $(q, \dot{q})$ space, whereas the variational integrators integrate the trajectory in the (q, p) space.

The strict comparison of the two integrators for trajectory optimization is usually difficult since it depends on a number of factors, such as the target problem, the optimizers used, the optimality and feasibility tolerances, etc. Therefore, we compare the Simpson variational integrator and the Hermite-Simpson direct collocation method by listing the order of accuracy, the number of variables and the number of constraints for trajectory optimization. In general, the computational loads of optimization depends on the problem size that is directly related with the number of variables and the the number of constraints. The higher-order accuracy suggests the possibility of large time steps in trajectory optimization, which reduces not only the problem size but the computational loads of optimization as well. The results are in Table 2.1.[2] It can be concluded that the Simpson variational integrator is more accurate and has less variables and constraints in trajectory optimization, especially for constrained mechanical systems.

Table 2.1. The comparison of the Simpson variational integrator with the Hermite-Simpson direct collocation method for trajectory optimization. The trajectory optimization problem has N stages and the mechanical system has n degrees of freedom, m holonomic constraints and is fully actuated with n control inputs. Note that all the integrators use three control points for integration.

Integrator	Accuracy	# of Variables	# of Constraints
Variational Integrator	4th-order	$(4N+3)n+(2N+1)m$	$3Nn+(2N+1)m$
Direct Collocation (Explicit)	3rd-order	$(6N+3)n+(2N+1)m$	$4Nn+(6N+3)m$
Direct Collocation (Implicit)	3rd-order	$(8N+4)n+(2N+1)m$	$(6N+1)n+(6N+3)m$

[2]The explicit and implicit formulations of the Hermite-Simpson direct collocation methods differ in whether the joint acceleration $\ddot{q}$ is explicitly computed or implicitly involved as extra variables. Although the explicit formulation of the Hermite-Simpson direct collocation has fewer variables and constraints than the implicit one, it is more complicated for the evaluation and linearization. Thus, the implicit formulation is more efficient and more commonly used in trajectory optimization [**27**].

The accuracy comparison in Table 2.1 of the Simpson variational integrator with the Hermite-Simpson direct collocation method is further numerically validated on a 12-link pendulum. In the comparison, different time steps are used to simulate 100 trajectories with the final time $T = 10$ s, and the initial joint angles q^0 are uniformly sampled from $[-\frac{\pi}{12}, \frac{\pi}{12}]$ and the initial joint velocities $\dot{q}^0$ are zero. Moreover, the Simpson variational integrator uses Algorithms 1 and 2 which has $O(n)$ complexity for the integrator evaluation and the Newton direction computation, whereas the Hermite-Simpson direct collocation method uses **[31,34]** which is $O(n)$ for the integrator evaluation and $O(n^3)$ for the Newton direction computation. For each initial condition, the benchmark solution $q_d(t)$ is created from the Hermite-Simpson direct collocation method with a time step of 5×10^{-4} s and the simulation error in $q(t)$ is evaluated as $\frac{1}{T}\text{int}_0^T \|q(t) - q_d(t)\| dt$. The running time of the simulation is also recorded. The results are in Fig. 2.3, which indicates that the Simpson variational integrator is more accurate and more efficient in simulation, and more importantly, a better alternative to the Hermite-Simpson direction collocation method for trajectory optimization.

In regard to the integrator evaluation and linearization, for unconstrained mechanical systems, experiments (not shown) suggest that the Simpson variational integrator using Algorithms 1, 4 and 5 is usually faster than the Hermite-Simpson direct collocation method using **[31, 34]** even though theoretically both integrators have the same order of complexity. However, for constrained mechanical systems, if there are m holonomic constraints, the Simpson variational integrator is $O(mn)$ for the evaluation and $O(mn^2)$ for

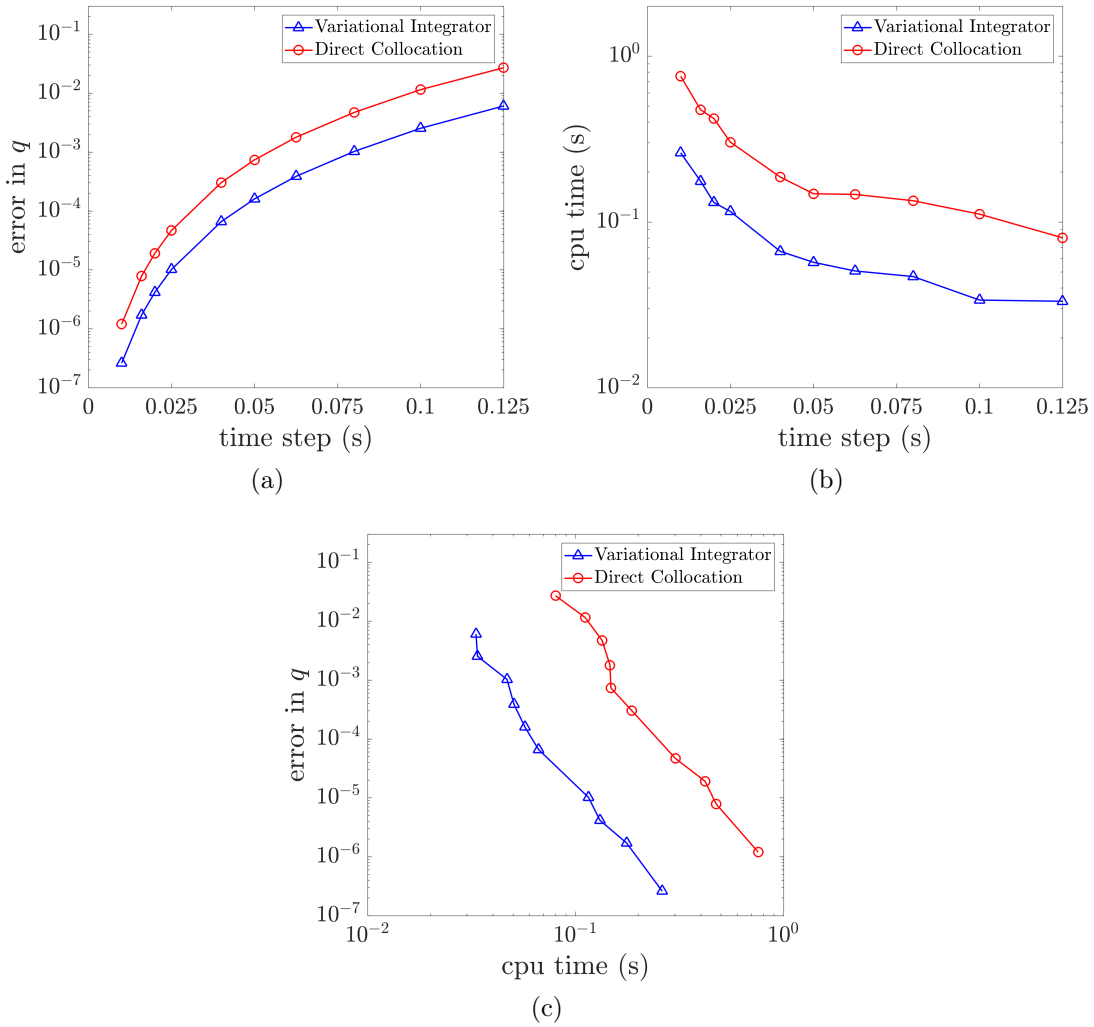


Figure 2.3. The comparison of the Simpson variational integrator with the Hermite-Simpson direction collocation method on a 12-link pendulum with different time steps. The results of the integrator error are in (a), the computational time in (b) and the integration error v.s. computational time in (c). Each result is calculated over 100 initial conditions.

the linearization while the Hermite-Simpson direct collocation method in [**26**,**27**] is respectively $O(mn^2)$ and $O(mn^3)$, the difference of which results from that the Hermite-Simpson direct collocation method is more complicated to model the constrained dynamics.

2.6. Trajectory Optimization

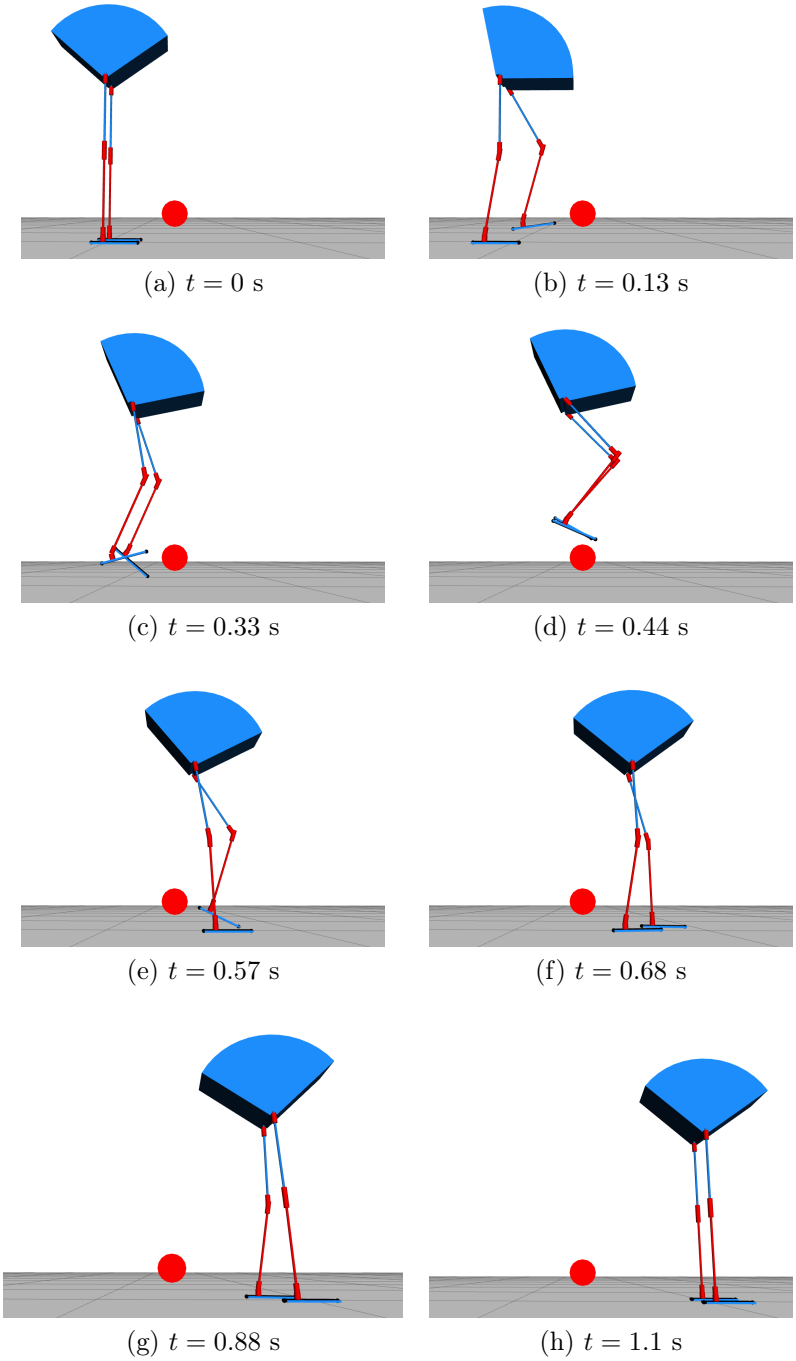
In this section, we implement the fourth-order Simpson variational integrator (Example 2.2) with Algorithms 1, 4 and 5 on the Spring Flamingo robot [**35**], the LittleDog robot [**36**] and the Atlas robot [**37**] for trajectory optimization, the results of which are included in our supplementary videos. It should be noted that the variational integrators used in [**19**–**21**, **23**, **25**] for trajectory optimization are second order. In Sections 2.6.1 and 2.6.2, a LCP formulation similar to [**25**] is used to model the discontinuous frictional contacts with which no contact mode needs to be prespecified. These examples indicate that higher-order variational integrators are good alternatives to the direct collocation methods [**26**, **27**]. The trajectory optimization problems are solved with SNOPT [**38**].

2.6.1. Spring Flamingo

The Spring Flamingo robot is a 9-DoF flat-footed biped robot with actuated hips and knees and passive springs at ankles [**35**]. In this example, the Spring Flamingo robot is commanded to jump over an obstacle that is 0.16 m high while walking horizontally from one position to another. The results are in Fig. 2.4, in which the initial walking velocity is 0.26 m/s and the average walking velocity is around 0.9 m/s.

2.6.2. LittleDog

The LittleDog robot is 18-DoF quadruped robot used in research of robot walking [**36**]. In this example, the LittleDog robot is required to walk over terrain with two gaps. The results are in Fig. 2.5, in which the average walking velocity is 0.25 m/s.



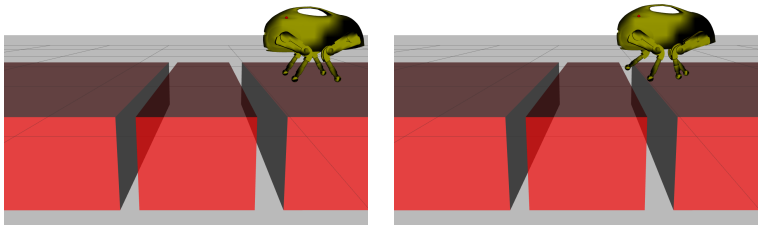
(a) $t = 0$ s (b) $t = 0.13$ s

(c) $t = 0.33$ s (d) $t = 0.44$ s

(e) $t = 0.57$ s (f) $t = 0.68$ s

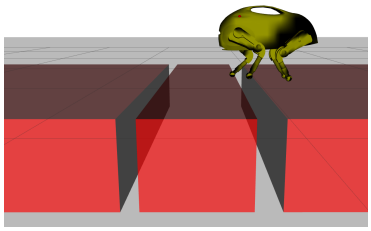
(g) $t = 0.88$ s (h) $t = 1.1$ s

Figure 2.4. The Spring Flamingo robot jumps over a obstacle of 0.16 meters high.

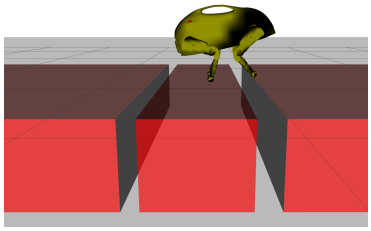


(a) $t = 0$ s

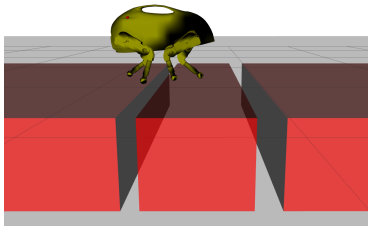
(b) $t = 0.48$ s



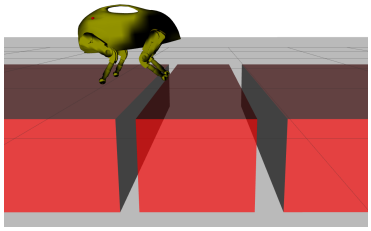
(c) $t = 0.56$ s



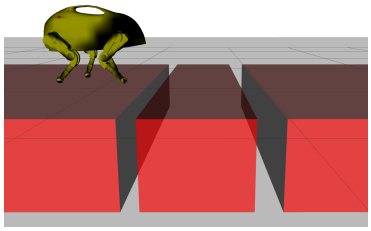
(d) $t = 1.04$ s



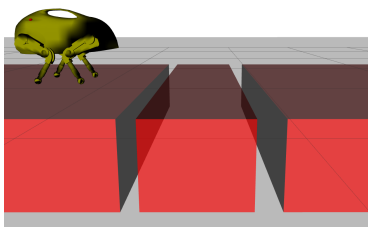
(e) $t = 1.84$ s



(f) $t = 2.16$ s



(g) $t = 2.72$ s

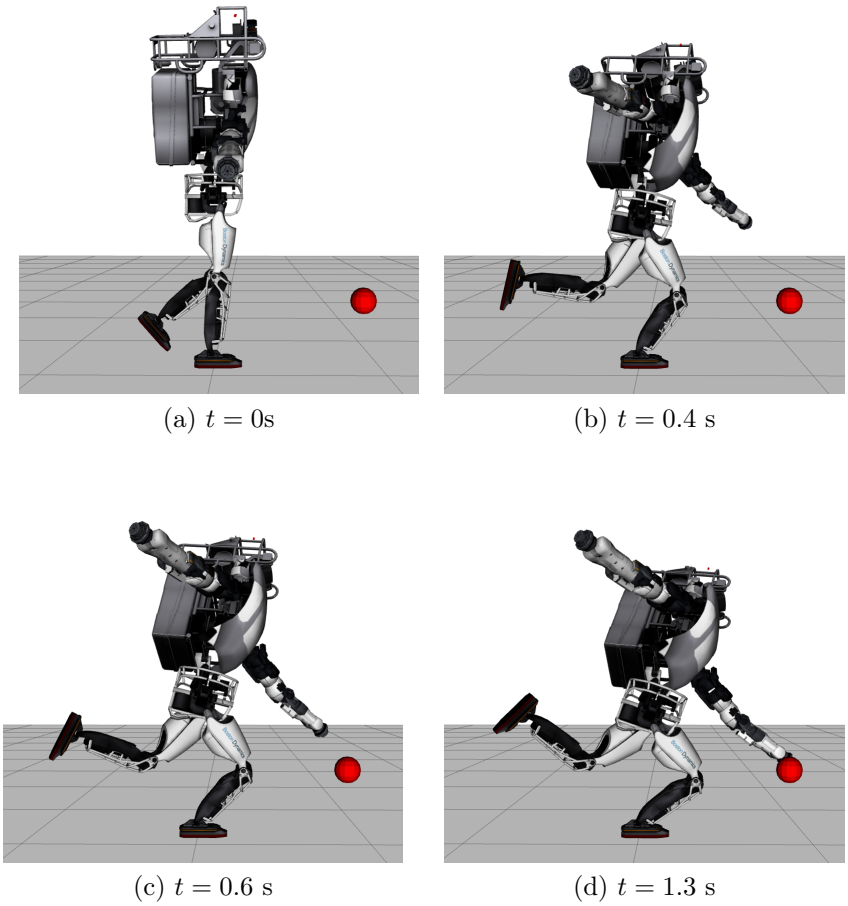


(h) $t = 3.2$ s

Figure 2.5. The LittleDog robot walks over terrain with gaps.

2.6.3. Atlas

The Atlas robot is a 30-DoF humanoid robot used in the DARPA Robotics Challenge [**37**]. In this example, the Atlas robot is required to pick a red ball with its left hand while keeping balanced only with its right foot. Moreover, the contact wrenches applied to the supporting foot should satisfy contact constraints of a flat foot [**27**]. The results are in Fig. 2.6 and it takes around 1.3 s for the Atlas robot to pick the ball.



(a) $t = 0$s (b) $t = 0.4$ s

(c) $t = 0.6$ s (d) $t = 1.3$ s

Figure 2.6. The Atlas robot picks a red ball while keeping balanced with a single foot.

2.7. Conclusion

In this chapter, we present $O(n)$ algorithms for the linear-time higher-order variational integrators and $O(n^2)$ algorithms to linearize the DEL equations for use in trajectory optimization. The proposed algorithms are validated through comparison with existing methods and implementation on robotic systems for trajectory optimization. The results illustrate that the same integrator can be used for simulation and trajectory optimization in robotics, preserving mechanical properties while achieving good scalability and accuracy. Though not presented in this chapter, these $O(n)$ algorithms can be regularized for parallel computation, which results in $O(\log(n))$ algorithms with enough processors. In particular, the proposed algorithms for higher-order variational integrators are expected to benefit existing optimal control and estimation techniques [**39**–**43**] in numerous aspects.

2.8. Proofs

In this section, we review and prove Propositions 2.1 to 2.4 although these proofs are not necessary for implementation.

2.8.1. Proof of Proposition 2.1

In Section 2.3.1, we define the discrete articulated body momentum and discrete articulated body impulse are respectively as follows.

Definition 2.4. The discrete articulated body momentum $\overline{\mu}_i^{k,\alpha} \in \mathbb{R}^6$ for articulated body i is defined to be

$$\overline{\mu}_i^{k,\alpha} = \overline{M}_i^{k,\alpha}\overline{v}_i^{k,\alpha} + \sum_{j \in \mathrm{chd}(i)} \overline{\mu}_j^{k,\alpha} \qquad \forall \alpha = 0,\, 1, \cdots,\, s \tag{2.35}$$

in which $\overline{M}_i^{k,\alpha}$ and $\overline{v}_i^{k,\alpha}$ are respectively the spatial inertia matrix and spatial velocity of rigid body i.

Definition 2.5. Suppose $\overline{F}_i(t) \in \mathbb{R}^6$ is the sum of all the wrenches directly acting on rigid body i, which does not include those applied or transmitted through the joints that are connected to rigid body i. The discrete articulated body impulse $H_i^{k,\alpha} \in \mathbb{R}^6$ for articulated body i is defined to be

$$H_i^{k,\alpha} = \overline{F}_i^{k,\alpha} + \sum_{j \in \mathrm{chd}(i)} H_j^{k,\alpha} \tag{2.36}$$

in which $\overline{F}_i^{k,\alpha} = \omega^\alpha \overline{F}_i(t^{k,\alpha})\Delta t \in \mathbb{R}^6$ is the discrete impulse acting on rigid body i. Note that $\overline{F}_i(t)$, $\overline{F}_i^{k,\alpha}$ and $H_i^{k,\alpha}$ are expressed in frame $\{0\}$.

The DEL equations Eq. (2.5) can be recursively evaluated with $\overline{\mu}_i^{k,\alpha}$ and $\overline{F}_i^{k,\alpha}$ as Proposition 2.5 indicates.

Proposition 2.5. If $Q_i(t) \in \mathbb{R}$ is the sum of all joint forces applied to joint i and $p^k = \begin{bmatrix} p_1^k & p_2^k & \cdots & p_n^k \end{bmatrix}^T \in \mathbb{R}^n$ is the discrete momentum, the DEL equations Eq. (2.5) can be evaluated as

$$r_i^{k,0} = p_i^k + \overline{S}_i^{k,0^T} \cdot \overline{\Omega}_i^{k,0} + \sum_{\beta=0}^{s} a^{0\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,0}, \tag{2.37a}$$

$$r_i^{k,\alpha} = \overline{S}_i^{k,\alpha^T} \cdot \overline{\Omega}_i^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,\alpha} \quad \forall \alpha = 1, \cdots, s-1, \tag{2.37b}$$

$$p_i^{k+1} = \overline{S}_i^{k,s^T} \cdot \overline{\Omega}_i^{k,s} + \sum_{\beta=0}^{s} a^{s\beta} \overline{S}_i^{k,\beta^T} \cdot \overline{\mu}_i^{k,\beta} + Q_i^{k,s} \tag{2.37c}$$

in which $r_i^{k,\alpha}$ is the residue of the DEL equations Eqs. (2.5a) and (2.5b), $a^{\alpha\beta} = w^\beta b^{\beta\alpha}$, $\overline{\Omega}_i^{k,\alpha} = w^\alpha \Delta t \cdot \mathrm{ad}^T_{\overline{v}_i^{k,\alpha}} \cdot \overline{\mu}_i^{k,\alpha} + H_i^{k,\alpha}$, and $Q_i^{k,\alpha} = \omega^\alpha Q_i(t^{k,\alpha})\Delta t$ is the discrete joint force applied to joint i.

Proof. The Lagrangian of a mechanical system is defined to be

$$\mathcal{L}(q, \dot{q}) = K(q, \dot{q}) - V(q) \tag{2.38}$$

in which $K(q, \dot{q})$ is the kinetic energy and $V(q)$ is the potential energy. It is by the definition of $\overline{F}_i(t)$ and $Q_i(t)$ that

$$\mathrm{int}_0^T \mathcal{F}(t) \cdot \delta q dt - \delta \mathrm{int}_0^T V(q) dt = \mathrm{int}_0^T \sum_{i=1}^n \overline{F}_i(t) \cdot \overline{\eta}_i dt + \mathrm{int}_0^T \sum_{i=1}^n Q_i(t) \cdot \delta q_i dt$$

in which $\overline{\eta}_i = (\delta g_i g_i^{-1})^\vee$. Therefore, the Lagrange-d'Alembert principle Eq. (2.1) is equivalent to

$$\delta \mathfrak{S} = \delta \mathrm{int}_0^T K(q, \dot{q}) dt + \mathrm{int}_0^T \sum_{i=1}^n \overline{F}_i(t) \cdot \overline{\eta}_i dt + \mathrm{int}_0^T \sum_{i=1}^n Q_i(t) \cdot \delta q_i dt = 0. \tag{2.39}$$

As a result of Eqs. (2.3) and (2.39), we have

$$\sum_{k=0}^{N-1} \sum_{\alpha=0}^{s} w^\alpha \sum_{i=1}^n \Big[\Big\langle \tfrac{\partial K}{\partial q_i}(q^{k,\alpha}, \dot{q}^{k,\alpha}), \delta q_i^{k,\alpha} \Big\rangle + \Big\langle \tfrac{\partial K}{\partial \dot{q}_i}(q^{k,\alpha}, \dot{q}^{k,\alpha}), \delta \dot{q}_i^{k,\alpha} \Big\rangle + \Big\langle \overline{F}_i(t^{k,\alpha}), \overline{\eta}_i^{k,\alpha} \Big\rangle + \big\langle Q_i(t^{k,\alpha}), \delta q_i^{k,\alpha} \big\rangle \Big] \Delta t = 0. \tag{2.40}$$

Note that the kinetic energy $K(q^{k,\alpha}, \dot{q}^{k,\alpha})$ is

$$K(q^{k,\alpha}, \dot{q}^{k,\alpha}) = \frac{1}{2} \sum_{j=1}^n \overline{v}_j^{k,\alpha T} \overline{M}_j^{k,\alpha} \overline{v}_j^{k,\alpha} \tag{2.41}$$

in which $\overline{M}_i^{k,\alpha} \in \mathbb{R}^{6\times 6}$ is the spatial inertia matrix and $\overline{v}_i^{k,\alpha} \in \mathbb{R}^6$ is the spatial velocity. Using Eqs. (2.16b), (2.35) and (2.41), we obtain

$$\begin{aligned}\frac{\partial K}{\partial \dot{q}_i}(q^{k,\alpha}, \dot{q}^{k,\alpha}) =& \sum_{j=1}^{n} \frac{\partial \overline{v}_j^{k,\alpha}{}^T}{\partial \dot{q}_i} \overline{M}_j^{k,\alpha} \overline{v}_j^{k,\alpha} \\ =& \overline{S}_i^{k,\alpha}{}^T \overline{M}_i^{k,\alpha} \overline{v}_i^{k,\alpha} + \sum_{j\in \mathrm{des}(i)} \overline{S}_i^{k,\alpha}{}^T \overline{M}_j^{k,\alpha} \overline{v}_j^{k,\alpha} \\ =& \overline{S}_i^{k,\alpha}{}^T \overline{\mu}_i^{k,\alpha}.\end{aligned} \tag{2.42}$$

In a similar way, as a result of Eqs. (2.17b), (2.18), (2.20b), (2.35) and (2.41), a tedious but straightforward algebraic manipulation results in

$$\begin{aligned}\frac{\partial K}{\partial q_i}(q^{k,\alpha}, \dot{q}^{k,\alpha}) =& \sum_{j\in \mathrm{des}(i)\cup\{i\}} \left[\mathrm{ad}_{\overline{S}_i^{k,\alpha}}(\overline{v}_j^{k,\alpha} - \overline{v}_i^{k,\alpha}) - \mathrm{ad}_{\overline{S}_i^{k,\alpha}} \overline{v}_j^{k,\alpha}\right]^T \overline{M}_j^{k,\alpha} v_j^{k,\alpha} \\ =& S_i^{k,\alpha}{}^T \mathrm{ad}_{\overline{v}_i}^T \cdot \overline{\mu}_i^{k,\alpha} \\ =& \dot{\overline{S}}_i^{k,\alpha}{}^T \overline{\mu}_i^{k,\alpha}.\end{aligned} \tag{2.43}$$

In addition, using Eqs. (2.10) and (2.36) and $\overline{F}_i^{k,\alpha} = w^\alpha \overline{F}_i(t^{k,\alpha})\Delta t$, we obtain

$$\begin{aligned}\sum_{i=1}^{n} \left\langle w^\alpha \overline{F}_i(t^{k,\alpha})\Delta t, \overline{\eta}_i^{k,\alpha}\right\rangle =& \sum_{i=1}^{n} \left\langle w^\alpha \overline{F}_i(t^{k,\alpha})\Delta t, \overline{S}_i^{k,\alpha} \cdot \delta q_i^{k,\alpha} + \sum_{j\in \mathrm{anc}(i)} \overline{S}_j^{k,\alpha} \cdot q_j^{k,\alpha}\right\rangle \\ =& \sum_{i=1}^{n} \left\langle \overline{F}_i^{k,\alpha} + \sum_{j\in \mathrm{des}(i)} \overline{F}_j^{k,\alpha}, \overline{S}_i^{k,\alpha} \cdot \delta q_i^{k,\alpha}\right\rangle \\ =& \sum_{i=1}^{n} \left\langle H_i^{k,\alpha}, \overline{S}_i^{k,\alpha} \cdot \delta q_i^{k,\alpha}\right\rangle \\ =& \sum_{i=1}^{n} \left\langle \overline{S}_i^{k,\alpha}{}^T H_i^{k,\alpha}, \delta q_j^{k,\alpha}\right\rangle.\end{aligned} \tag{2.44}$$

From Eq. (2.2), we obtain

$$\delta\dot{q}_i^{k,\alpha} = \frac{1}{\Delta t}\sum_{\beta=0}^{s} b^{\alpha\beta}\cdot\delta q_i^{k,\beta}. \tag{2.45}$$

Substituting Eqs. (2.42) to (2.44) into Eq. (2.40) and simplifying the resulting equation with Eq. (2.45) as well as the chain rule, we obtain

$$\sum_{k=0}^{N-1}\sum_{\alpha=0}^{s}\sum_{i=1}^{n}\left\langle \overline{S}_i^{k,\alpha T}\cdot\overline{\Omega}_i^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta}\overline{S}_i^{k,\beta T}\cdot\overline{\mu}_i^{k,\beta} + Q_i^{k,\alpha}, \delta q_i^{k,\alpha}\right\rangle = 0$$

in which $a^{\alpha\beta} = w^\beta b^{\beta\alpha}$, $\overline{\Omega}_i^{k,\alpha} = w^\alpha \Delta t\cdot \mathrm{ad}^T_{\overline{v}_i^{k,\alpha}}\cdot\overline{\mu}_i^{k,\alpha} + H_i^{k,\alpha}$ and $Q_i^{k,\alpha} = \omega^\alpha Q_i(t^{k,\alpha})\Delta t$. The equation above is equivalent to requiring

$$p_i^k + \overline{S}_i^{k,0T}\cdot\overline{\Omega}_i^{k,0} + \sum_{\beta=0}^{s} a^{0\beta}\overline{S}_i^{k,\beta T}\cdot\overline{\mu}_i^{k,\beta} + Q_i^{k,0} = 0,$$

$$\overline{S}_i^{k,\alpha T}\cdot\overline{\Omega}_i^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta}\overline{S}_i^{k,\beta T}\cdot\overline{\mu}_i^{k,\beta} + Q_i^{k,\alpha} = 0 \quad \forall\alpha = 1,\cdots,s-1,$$

$$p_i^{k+1} = \overline{S}_i^{k,sT}\cdot\overline{\Omega}_i^{k,s} + \sum_{\beta=0}^{s} a^{s\beta}\overline{S}_i^{k,\beta T}\cdot\overline{\mu}_i^{k,\beta} + Q_i^{k,s}.$$

This completes the proof. □

2.8.2. Proof of Proposition 2.2

In Section 2.3.2, we make the assumption on the discrete impulse $\overline{F}_i^{k,\alpha}$ and discrete joint force $Q_i^{k,\alpha}$ as follows.

Assumption 2.1. Let $u(t)$ be control inputs of the mechanical system, we assume that the discrete impulse $\overline{F}_i^{k,\alpha}$ and discrete joint force $Q_i^{k,\alpha}$ can be respectively formulated as $\overline{F}_i^{k,\alpha} = \overline{F}_i^{k,\alpha}(g_i^{k,\alpha}, \overline{v}_i^{k,\alpha}, u^{k,\alpha})$ and $Q_i^{k,\alpha} = Q_i^{k,\alpha}(q_i^{k,\alpha}, \dot{q}_i^{k,\alpha}, u^{k,\alpha})$ in which $u^{k,\alpha} = u(t^{k,\alpha})$.

From the notion of the spatial variation in Section 2.2.5, we have the following proposition for the Newton direction computation, which is later used in the proof of Proposition 2.2.

Proposition 2.8.1. If $\delta q_i^{k,\alpha}$ is the Newton direction for $q_i^{k,\alpha}$, $r_i^{k,\alpha}$ is the residue of the DEL equations Eqs. (2.31a) and (2.31b), and Assumption 2.1 holds, the computation of the Newton direction $\delta q_i^{k,\alpha}$ is equivalent to requiring

$$\overline{\delta}\overline{\mu}_i^{k,\alpha} = \overline{M}_i^{k,\alpha}\overline{\delta}\overline{v}_i^{k,\alpha} + \sum_{j\in\mathrm{chd}(i)} \left(\overline{\delta}\overline{\mu}_j^{k,\alpha} - \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} \cdot \delta q_j^{k,\alpha}\right), \quad \forall \alpha = 0,\, 1,\, \cdots,\, s, \tag{2.46a}$$

$$\begin{aligned}\overline{\delta}\, H_i^{k,\alpha} = \left(\mathbb{D}_1 \overline{F}_i^{k,\alpha} + \mathrm{ad}^D_{\overline{F}_i^{k,\alpha}} - \mathrm{ad}_{\overline{v}_i^{k,\alpha}}\right) \cdot \overline{\eta}_i^{k,\alpha} + \mathbb{D}_2 \overline{F}_i^{k,\alpha} \cdot \overline{\delta}\overline{v}_i^{k,\alpha} + \\ \sum_{j\in\mathrm{chd}(i)} \left(\overline{\delta}\, H_j^{k,\alpha} - \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} \cdot \delta q_j^{k,\alpha}\right), \quad \forall \alpha = 0,\, 1,\, \cdots,\, s-1,\end{aligned} \tag{2.46b}$$

$$\overline{\delta}\,\overline{\Omega}_i^{k,\alpha} = \omega^\alpha \Delta t \cdot \left(\mathrm{ad}^T_{\overline{v}_i^{k,\alpha}} \cdot \overline{\delta}\overline{\mu}_i^{k,\alpha} + \mathrm{ad}^D_{\overline{\mu}_i^{k,\alpha}} \overline{\delta}\overline{v}_i^{k,\alpha}\right) + \overline{\delta}\, H_i^{k,\alpha}, \quad \forall \alpha = 0,\, 1,\, \cdots,\, s-1, \tag{2.46c}$$

$$\begin{aligned}\overline{S}_i^{k,\alpha^T} \overline{\delta}\,\overline{\Omega}_i^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_i^{k,\beta^T} \overline{\delta}\overline{\mu}_i^{k,\beta} + \mathbb{D}_1 Q_i^{k,\alpha} \cdot \delta q_i^{k,\alpha} + \\ \mathbb{D}_2 Q_i^{k,\alpha} \cdot \delta \dot{q}_i^{k,\alpha} = -r_i^{k,\alpha}, \quad \forall \alpha = 0,\, 1,\, \cdots,\, s-1.\end{aligned} \tag{2.46d}$$

in which $\overline{\delta}\overline{v}_i^{k,\alpha}$, $\overline{\delta}\overline{\mu}_i^{k,\alpha}$, $\overline{\delta}\,H_i^{k,\alpha}$ and $\overline{\delta}\,\overline{\Omega}_i^{k,\alpha}$ are the spatial variations of $\overline{v}_i^{k,\alpha}$, $\overline{\mu}_i^{k,\alpha}$, $H_i^{k,\alpha}$ and $\overline{\Omega}_i^{k,\alpha}$, respectively. Note that $\delta q_i^{k,0} = 0$ and $\overline{\eta}_i^{k,0} = 0$ though $\overline{\delta}\overline{v}_i^{k,0} \neq 0$.

Proof. Eqs. (2.46a) and (2.46c) are respectively the same as Eqs. (2.28) and (2.30), thus we only need to prove Eqs. (2.46b) and (2.46d).

From Assumption 2.1, we have $\overline{F}_i^{k,\alpha} = \overline{F}_i^{k,\alpha}(g_i^{k,\alpha}, \overline{v}_i^{k,\alpha}, u^{k,\alpha})$, and since $\delta u_i^{k,\alpha} = 0$, we obtain $\delta\overline{F}_i^{k,\alpha}$ as

$$\delta\overline{F}_i^{k,\alpha} = \mathbb{D}_1\overline{F}_i^{k,\alpha} \cdot \overline{\eta}_i^{k,\alpha} + \mathbb{D}_2\overline{F}_i^{k,\alpha} \cdot \delta\overline{v}_i^{k,\alpha}.$$

According to Eq. (2.22), the spatial variation $\overline{\delta}\,\overline{F}_i^{k,\alpha}$ is

$$\overline{\delta}\,\overline{F}_i^{k,\alpha} = \mathbb{D}_1\overline{F}_i^{k,\alpha} \cdot \overline{\eta}_i^{k,\alpha} + \mathbb{D}_2\overline{F}_i^{k,\alpha} \cdot \delta\overline{v}_i^{k,\alpha} + \mathrm{ad}^T_{\overline{\eta}_i^{k,\alpha}}\overline{F}_i^{k,\alpha}.$$

Since $\delta\overline{v}_i^{k,\alpha} = \overline{\delta}\overline{v}_i^{k,\alpha} + \mathrm{ad}_{\overline{\eta}_i^{k,\alpha}}\overline{v}_i^{k,\alpha}$, $\mathrm{ad}_{\overline{v}_i^{k,\alpha}}\overline{\eta}_i^{k,\alpha} = -\mathrm{ad}_{\overline{\eta}_i^{k,\alpha}}\overline{v}_i^{k,\alpha}$ as well as $\mathrm{ad}^T_{\overline{\eta}_i^{k,\alpha}}\overline{F}_i^{k,\alpha} = \mathrm{ad}^D_{\overline{F}_i^{k,\alpha}}\overline{\eta}_i^{k,\alpha}$, the equation above is equivalent to

$$\overline{\delta}\,\overline{F}_i^{k,\alpha} = \big(\mathbb{D}_1\overline{F}_i^{k,\alpha} + \mathrm{ad}^D_{\overline{F}_i^{k,\alpha}} - \mathbb{D}_2\overline{F}_i^{k,\alpha}\mathrm{ad}_{\overline{v}_i^{k,\alpha}}\big) \cdot \overline{\eta}_i^{k,\alpha} + \mathbb{D}_2\overline{F}_i^{k,\alpha} \cdot \overline{\delta}\overline{v}_i^{k,\alpha}.$$

Substitute the equation above into Eq. (2.29), the result of which is Eq. (2.46b).

As for the proof of Eq. (2.46d), from Eqs. (2.31a) and (2.31b), the Newton direction $\delta q_i^{k,\alpha}$ requires that

$$\delta\big(S_i^{k,\alpha^T}\overline{\Omega}_i\big) + \sum_{\beta=0}^{s} a^{\alpha\beta}\delta\big(S_i^{k,\beta^T}\overline{\mu}_i^{k,\beta}\big) + \mathbb{D}_1 Q_i^{k,\alpha} \cdot \delta q_i^{k,\alpha} + \mathbb{D}_2 Q_i^{k,\alpha} \cdot \delta\dot{q}_i^{k,\alpha} = -r_i^{k,\alpha} \quad \forall\alpha = 0,\, 1,\, \cdots,\, s-1. \tag{2.47}$$

As a result of Eqs. (2.23) and (2.24), we have $\delta\big(\overline{S}_i^{k,\alpha^T}\overline{\mu}_i^{k,\alpha}\big) = \overline{S}_i^{k,\alpha^T}\overline{\delta}\overline{\mu}_i^{k,\alpha}$ and $\delta\big(\overline{S}_i^{k,\alpha^T}\overline{\Omega}_i^{k,\alpha}\big) = \overline{S}_i^{k,\alpha^T}\overline{\delta}\,\overline{\Omega}_i^{k,\alpha}$, with which and Eq. (2.47), we obtain Eq. (2.46d). This completes the proof. □

In Section 2.3.2, Proposition 2.2 to compute the Newton direction is stated as follows, for which note that the higher-order variational integrator has $s+1$ control points and the mechanical system has n degrees of freedom.

Proposition 2.8.2. For higher-order variational integrators of unconstrained mechanical systems, if Assumption 2.1 holds and $\mathcal{J}^{k^{-1}}(\overline{q}^k)$ exists, the Newton direction $\delta\overline{q}^k = -\mathcal{J}^{k^{-1}}(\overline{q}^k)\cdot r^k$ can be computed with Algorithm 2 in $O(s^3n)$ time.

Proof. The proof consists of proving the correctness and the $O(n)$ complexity of the algorithms.

For each $j \in \mathrm{chd}(i)$, we suppose that there exists $D_j^{k,\alpha\rho}$, $G_j^{k,\alpha\nu}$, $l_j^{k,\alpha}$ and $\Pi_j^{k,\alpha\rho}$, $\Psi_j^{k,\alpha\nu}$, $\zeta_j^{k,\alpha}$ such that

$$\overline{\delta}\overline{\mu}_j^{k,\alpha} = \sum_{\rho=0}^{s} D_j^{k,\alpha\rho}\cdot\overline{\delta}\overline{v}_j^{k,\rho} + \sum_{\nu=1}^{s} G_j^{k,\alpha\nu}\cdot\overline{\eta}_j^{k,\nu} + l_j^{k,\alpha},\quad \forall\alpha = 0,\,1,\,\cdots,\,s, \tag{2.48}$$

$$\overline{\delta}\,H_j^{k,\alpha} = \sum_{\rho=0}^{s} \Pi_j^{k,\alpha\rho}\cdot\overline{\delta}\overline{v}_j^{k,\rho} + \sum_{\nu=1}^{s} \Psi_j^{k,\alpha\nu}\cdot\overline{\eta}_j^{k,\nu} + \zeta_j^{k,\alpha},\quad \forall\alpha = 0,\,1,\,\cdots,\,s-1. \tag{2.49}$$

According to Eqs. (2.9), (2.25) and (2.45), $\overline{\delta}\overline{v}_j^{k,\rho}$ and $\overline{\eta}_j^{k,\nu}$ can be respectively computed as

$$\overline{\eta}_j^{k,\nu} = \overline{\eta}_i^{k,\nu} + \overline{S}_j^{k,\nu}\cdot\delta q_j^{k,\nu} \tag{2.50}$$

and

$$\overline{\delta}\overline{v}_j^{k,\rho} = \overline{\delta}\overline{v}_i^{k,\rho} + \dot{\overline{S}}_j^{k,\rho} \cdot \delta q_j^{k,\rho} + \frac{1}{\Delta t}\overline{S}_j^{k,\rho} \sum_{\gamma=1}^{s} b^{\rho\gamma} \cdot \delta q_j^{k,\gamma} \tag{2.51}$$

for which note that $\delta q_j^{k,0} = 0$. Substitute Eqs. (2.50) and (2.51) into Eq. (2.48), algebraic manipulation shows that

$$\overline{\delta}\overline{\mu}_j^{k,\alpha} = \sum_{\rho=0}^{s} D_j^{k,\alpha\rho} \cdot \overline{\delta}\overline{v}_i^{k,\rho} + \sum_{\nu=1}^{s} G_j^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + l_j^{k,\alpha} + \sum_{\gamma=1}^{s} H_j^{k,\alpha\gamma} \delta q_j^{k,\gamma}, \tag{2.52}$$

in which

$$H_j^{k,\alpha\gamma} = D_j^{k,\alpha\gamma} \dot{\overline{S}}_j^{k,\gamma} + G_j^{k,\alpha\gamma} \overline{S}_j^{k,\gamma} + \frac{1}{\Delta t} \sum_{\rho=0}^{s} b^{\rho\gamma} D_j^{k,\alpha\rho} \overline{S}_j^{k,\rho}.$$

In a similar way, using Eqs. (2.49) to (2.51), we also have

$$\overline{\delta}\, H_j^{k,\alpha} = \sum_{\rho=0}^{s} \Pi_j^{k,\alpha\rho} \cdot \overline{\delta}\overline{v}_i^{k,\rho} + \sum_{\nu=1}^{s} \Psi_j^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + \zeta^{k,\alpha} + \sum_{\gamma=1}^{s} \Phi_j^{k,\alpha\gamma} \delta q_j^{k,\gamma} \tag{2.53}$$

in which

$$\Phi_j^{k,\alpha\gamma} = \Pi_j^{k,\alpha\gamma} \dot{\overline{S}}_j^{k,\gamma} + \Psi_j^{k,\alpha\gamma} \overline{S}_j^{k,\gamma} + \frac{1}{\Delta t} \sum_{\rho=0}^{s} b^{\rho\gamma} \Pi_j^{k,\alpha\rho} \overline{S}_j^{k,\rho}.$$

From Eqs. (2.18), (2.30) and (2.51) to (2.53) and

$$\overline{S}_j^{k,\alpha^T} \mathrm{ad}^T_{\overline{S}_j^{k,\alpha}} \overline{\mu}_j^{k,\alpha} = \overline{S}_j^{k,\alpha^T} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} = 0,$$

we obtain

$$\overline{S}_j^{k,\alpha^T} \overline{\delta}\,\overline{\Omega}_j^{k,\alpha} = \sum_{\rho=0}^{s} \Theta_j^{k,\alpha\rho} \cdot \overline{\delta}\overline{v}_i^{k,\rho} + \sum_{\nu=1}^{s} \Xi^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + \xi_j^{k,\alpha} \tag{2.54}$$

in which

$$\Theta_j^{k,\alpha\rho} = w^\alpha \Delta t \cdot \left(\dot{\overline{S}}_j^{k,\alpha^T} D_j^{k,\alpha\rho} + \sigma^{\alpha\rho} \overline{S}_j^{k,\alpha^T} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}}\right) + \overline{S}_j^{k,\alpha^T} \Pi_j^{k,\alpha\rho},$$

$$\Xi_j^{k,\alpha\nu} = w^\alpha \Delta t \cdot \dot{\overline{S}}_j^{k,\alpha^T} G_j^{k,\alpha\nu} + \overline{S}_j^{k,\alpha^T} \Psi_j^{k,\alpha\nu},$$

$$\begin{aligned}\xi_j^{k,\alpha} &= w^\alpha \Delta t \cdot \dot{\overline{S}}_j^{k,\alpha^T} l_j^{k,\alpha} + \overline{S}_j^{k,\alpha^T} \zeta_j^{k,\alpha} + \sum_{\gamma=1}^{s} \Big[w^\alpha \Delta t \cdot \left(\dot{\overline{S}}_j{}^{k,\alpha^T} H_j^{k,\alpha\gamma} + \right. \\ &\qquad \left. \sigma^{\alpha\gamma} \overline{S}_j^{k,\alpha^T} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \dot{\overline{S}}_j^{k,\alpha}\right) + \overline{S}_j^{k,\alpha^T} \Phi_j^{k,\alpha\gamma} \Big] \delta q_j^{k,\gamma},\end{aligned}$$

and note that $\sigma^{\alpha\rho}$ is given in Eq. (2.33) of Algorithm 3. Substituting Eqs. (2.45), (2.52) and (2.54) into Eq. (2.46d), we obtain

$$\sum_{\rho=0}^{s} \overline{\Theta}_j^{k,\alpha\rho} \cdot \bar{\delta}\overline{v}_i^{k,\rho} + \sum_{\nu=1}^{s} \overline{\Xi}_j^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + \overline{\xi}_j^{k,\alpha} + \sum_{\gamma=1}^{s} \Lambda_j^{k,\alpha\gamma} \cdot \delta q_j^{k,\gamma} = -r_j^{k,\alpha}, \tag{2.55}$$

$$\forall \alpha = 0,\, 1,\, \cdots,\, s-1.$$

in which

$$\overline{\Theta}_j^{k,\alpha\rho} = \Theta_j^{k,\alpha\rho} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_j^{k,\beta^T} D_j^{k,\beta\rho},$$

$$\overline{\Xi}_j^{k,\alpha\nu} = \Xi_j^{k,\alpha\nu} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_j^{k,\beta^T} G_j^{k,\beta\nu},$$

$$\overline{\xi}_j^{k,\alpha} = w^\alpha \Delta t \cdot \dot{\overline{S}}_j^{k,\alpha^T} l_j^{k,\alpha} + \overline{S}_j^{k,\alpha^T} \zeta_j^{k,\alpha} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_j^{k,\beta^T} l_j^{k,\beta},$$

$$\begin{aligned}\Lambda_j^{k,\alpha\gamma} =& w^\alpha \Delta t \cdot \dot{\overline{S}}_j^{k,\alpha^T} H_j^{k,\alpha\gamma} + \overline{S}_j^{k,\alpha^T} \Phi_j^{k,\alpha\gamma} + \sum_{\beta=0}^{s} a^{\alpha\beta} \overline{S}_j^{k,\beta^T} H_j^{k,\beta\gamma} + \\ &\sigma^{\alpha\gamma} \left(\mathbb{D}_1 Q_j^{k,\alpha} + w^\alpha \Delta t \cdot \overline{S}_j^{k,\alpha^T} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \dot{\overline{S}}_j^{k,\alpha} \right) + \frac{1}{\Delta t} b^{\alpha\gamma} \cdot \mathbb{D}_2 Q_j^{k,\alpha}.\end{aligned}$$

For notational convenience, we define $\Delta_j^{k,\alpha}$ to be

$$\Delta_j^{k,\alpha} = \sum_{\rho=0}^{s} \overline{\Theta}_j^{k,\alpha\rho} \cdot \overline{\delta v}_i^{k,\rho} + \sum_{\nu=1}^{s} \overline{\Xi}_j^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + \overline{\xi}_j^{k,\alpha}, \quad \forall \alpha = 0,\, 1,\, \cdots,\, s-1. \tag{2.56}$$

such that Eq. (2.55) is rewritten as

$$\sum_{\gamma=1}^{s} \Lambda_j^{k,\alpha\gamma} \cdot \delta q_j^{k,\gamma} = -r_j^{k,\alpha} - \Delta_j^{k,\alpha}, \quad \forall \alpha = 0,\, 1,\, \cdots,\, s-1. \tag{2.57}$$

In addition, if we further define Λ_j^k, r_j^k, Δ_j^k and $\delta\overline{q}_j^k$ respectively as

$$\begin{aligned}
\Lambda_j^k &= \left[\Lambda_j^{k,\alpha\gamma}\right] \in \mathbb{R}^{s\times s}, \\
r_j^k &= \begin{bmatrix} r_j^{k,0} & r_j^{k,1} & \cdots & r_j^{k,s-1} \end{bmatrix}^T \in \mathbb{R}^s, \\
\Delta_j^k &= \begin{bmatrix} \Delta_j^{k,0} & \Delta_j^{k,1} & \cdots & \Delta_j^{k,s-1} \end{bmatrix}^T \in \mathbb{R}^s, \\
\delta\overline{q}_j^k &= \begin{bmatrix} \delta q_j^{k,1} & \delta q_j^{k,2} & \cdots & \delta q_j^{k,s} \end{bmatrix}^T \in \mathbb{R}^s,
\end{aligned}$$

in which $0 \le \alpha \le s-1$ and $1 \le \gamma \le s$, then Eq. (2.57) is equivalent to requiring

$$\Lambda_j^k \cdot \delta\overline{q}_j^k = -r_j^k - \Delta_j^k. \tag{2.58}$$

in which Λ_j^k is invertible since $\mathcal{J}^{k^{-1}}(\overline{q}^k)$ exists. From Eq. (2.58), we obtain

$$\delta\overline{q}_j^k = -\Lambda_j^{k^{-1}}(r_j^k + \Delta_j^k).$$

If $\Lambda_j^{k^{-1}}$ is explicitly written as $\Lambda_j^{k^{-1}} = \left[\overline{\Lambda}_j^{k,\gamma\varrho}\right] \in \mathbb{R}^{s\times s}$ in which $1 \leq \gamma \leq s$ and $0 \leq \varrho \leq s-1$, expanding the equation above, we obtain

$$\delta q_j^{k,\gamma} = -\sum_{\varrho=0}^{s-1} \overline{\Lambda}_j^{k,\gamma\varrho} \left(r_j^{k,\varrho} + \Delta_j^{k,\varrho} \right), \quad \forall \gamma = 1,\, 2,\, \cdots,\, s. \tag{2.59}$$

Substitute Eq. (2.56) into Eq. (2.59), the result is

$$\delta q_j^{k,\gamma} = \sum_{\rho=0}^{s} X_j^{k,\gamma\rho} \cdot \overline{\delta v}_i^{k,\rho} + \sum_{\nu=1}^{s} Y_j^{k,\gamma\nu} \cdot \overline{\eta}_i^{k,\nu} + y_j^{k,\gamma} \tag{2.60}$$

in which

$$\begin{aligned} X_j^{k,\gamma\rho} &= -\sum_{\varrho=0}^{s-1} \overline{\Lambda}_j^{k,\gamma\varrho} \cdot \overline{\Theta}_j^{k,\varrho\rho}, \\ Y_j^{k,\gamma\nu} &= -\sum_{\varrho=0}^{s-1} \overline{\Lambda}_j^{k,\gamma\varrho} \cdot \overline{\Xi}_j^{k,\varrho\nu}, \\ y_j^{k,\gamma} &= -\sum_{\varrho=0}^{s-1} \overline{\Lambda}_j^{k,\gamma\varrho} \left(r_j^{k,\varrho} + \overline{\xi}_j^{k,\varrho} \right). \end{aligned}$$

Making use of Eqs. (2.52) and (2.60) and canceling out $\delta q_j^{k,\gamma}$, we obtain

$$\overline{\delta}\overline{\mu}_j^{k,\alpha} - \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} \cdot \delta q_j^{k,\alpha} = \sum_{\rho=0}^{s} \overline{D}_j^{k,\rho} \cdot \overline{\delta}\overline{v}_i^{k,\rho} + \sum_{\nu=1}^{s} \overline{G}_j^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + \overline{l}_j^{k,\alpha} \tag{2.61}$$

in which $\alpha = 0,\, 1,\, \cdots,\, s$, and

$$\overline{D}_j^{k,\rho} = D_j^{k,\rho} + \sum_{\gamma=1}^{s} H_j^{k,\alpha\gamma} X_j^{k,\gamma\rho} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} X_j^{k,\alpha\rho}, \tag{2.62a}$$

$$\overline{G}_j^{k,\nu} = G_j^{k,\alpha\nu} + \sum_{\gamma=1}^{s} H_j^{k,\alpha\gamma} Y_j^{k,\gamma\nu} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} Y_j^{k,\alpha\nu}, \tag{2.62b}$$

$$\overline{l}_j^{k,\alpha} = l_j^{k,\alpha} + \sum_{\gamma=1}^{s} H_j^{k,\alpha\gamma} y_j^{k,\gamma} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{\overline{\mu}_j^{k,\alpha}} \overline{S}_j^{k,\alpha} y_j^{k,\alpha}, \tag{2.62c}$$

and note that $\overline{\sigma}^{\alpha 0}$ is given in Eq. (2.33) of Algorithm 3. In a similar way, using Eqs. (2.53) and (2.60), we obtain

$$\overline{\delta}\, H_j^{k,\alpha} - \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} \cdot \delta q_j^{k,\alpha} = \sum_{\rho=0}^{s} \overline{\Pi}_j^{k,\alpha\rho} \cdot \overline{\delta} \overline{v}_j^{k,\rho} + \sum_{\nu=1}^{s} \overline{\Psi}_j^{k,\alpha\nu} \cdot \overline{\eta}_j^{k,\nu} + \overline{\zeta}_j^{k,\alpha} \tag{2.63}$$

in which $\alpha = 1,\, 2,\, \cdots,\, s$, and

$$\overline{\Pi}_j^{k,\alpha\rho} = \Pi_j^{k,\alpha\rho} + \sum_{\gamma=1}^{s} \Phi_j^{k,\alpha\gamma} X_j^{k,\gamma\rho} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} X_j^{k,\alpha\rho}, \tag{2.64a}$$

$$\overline{\Psi}_j^{k,\alpha\nu} = \Psi_j^{k,\alpha\nu} + \sum_{\gamma=1}^{s} \Phi_j^{k,\alpha\gamma} Y_j^{k,\gamma\nu} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} Y_j^{k,\alpha\nu}, \tag{2.64b}$$

$$\overline{\zeta}_j^{k,\alpha} = \zeta_j^{k,\alpha} + \sum_{\gamma=1}^{s} \Phi_j^{k,\alpha\gamma} y_j^{k,\gamma} - \overline{\sigma}^{\alpha 0} \mathrm{ad}^D_{H_j^{k,\alpha}} \overline{S}_j^{k,\alpha} y_j^{k,\alpha}. \tag{2.64c}$$

Finally, for each $j \in \mathrm{chd}(i)$, substituting Eqs. (2.61) and (2.63) respectively into Eqs. (2.46a) and (2.46b) and applying Eqs. (2.62) and (2.64) to expand $\overline{D}_j^{k,\rho}$, $\overline{G}_j^{k,\nu}$, $\overline{l}_j^{k,\alpha}$ and $\overline{\Pi}_j^{k,\alpha\rho}$, $\overline{\Psi}_j^{k,\alpha\nu}$, $\overline{\zeta}_j^{k,\alpha}$, we respectively obtain $D_i^{k,\rho}$, $G_i^{k,\nu}$, $l_i^{k,\alpha}$ and $\Pi_i^{k,\alpha\rho}$, $\Psi_i^{k,\alpha\nu}$, $\zeta_i^{k,\alpha}$ as Eqs. (2.32) and (2.34) of Algorithm 3 such that

$$\overline{\delta} \overline{\mu}_i^{k,\alpha} = \sum_{\rho=0}^{s} D_i^{k,\alpha\rho} \cdot \overline{\delta} \overline{v}_i^{k,\rho} + \sum_{\nu=1}^{s} G_i^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + l_i^{k,\alpha}, \qquad \forall \alpha = 0,\, 1,\, \cdots,\, s, \tag{2.65}$$

$$\overline{\delta}\, H_i^{k,\alpha} = \sum_{\rho=0}^{s} \Pi_i^{k,\alpha\rho} \cdot \overline{\delta}\overline{v}_i^{k,\rho} + \sum_{\nu=1}^{s} \Psi_i^{k,\alpha\nu} \cdot \overline{\eta}_i^{k,\nu} + \zeta_i^{k,\alpha}, \qquad \forall \alpha = 0,\, 1,\, \cdots,\, s-1. \tag{2.66}$$

In particular, note that even if rigid body i is the leaf node of the tree representation whose chd$(i) =$, there still exists $D_i^{k,\rho}$, $G_i^{k,\nu}$, $l_i^{k,\alpha}$ and $\Pi_i^{k,\alpha\rho}$, $\Psi_i^{k,\alpha\nu}$, $\zeta_i^{k,\alpha}$ from Eqs. (2.32) and (2.34) of Algorithm 3. Moreover, as long as $D_i^{k,\rho}$, $G_i^{k,\nu}$, $l_i^{k,\alpha}$ and $\Pi_i^{k,\alpha\rho}$, $\Psi_i^{k,\alpha\nu}$, $\zeta_i^{k,\alpha}$ are given for each rigid body i, we can further obtain $X_i^{k,\alpha\rho}$, $Y_i^{k,\alpha\nu}$, $y_i^{k,\alpha}$ following lines 3 to 9 of Algorithm 3.

In summary, for each rigid body i, we have shown that $X_i^{k,\alpha\rho}$, $Y_i^{k,\alpha\nu}$, $y_i^{k,\alpha}$ as well as $D_i^{k,\rho}$, $G_i^{k,\nu}$, $l_i^{k,\alpha}$ and $\Pi_i^{k,\alpha\rho}$, $\Psi_i^{k,\alpha\nu}$, $\zeta_i^{k,\alpha}$ are computable through the backward pass by Algorithm 3, and $\delta q_i^{k,\alpha}$ as well as $\overline{\eta}_i^{k,\alpha}$ and $\overline{\delta}\overline{v}_i^{k,\alpha}$ are computable through the forward pass by lines 4 to 15 of Algorithm 2, which proves the correctness of the algorithms.

In regard to the complexity, Algorithm 3 has $O(s^2)+O(s^3)$ complexity since there are $O(s^2)$ quantities and the computation of ${\Lambda_i^{k,\alpha}}^{-1}$ takes $O(s^3)$ time, and thus the backward pass by lines 1 to 3 of Algorithm 2 totally takes $O(s^3n+s^2n)$ time. Moreover, in lines 4 to 15 of Algorithm 2, the forward pass takes $O(s^2n)$ time. As a result, the overall complexity of Algorithm 2 is $O(s^3n)$, which proves the complexity of the algorithms. □

2.8.3. Proof of Proposition 2.3

Proposition 2.8.3. For the kinetic energy $K(q,\dot{q})$ of a mechanical system, $\frac{\partial^2 K}{\partial \dot{q}^2}$, $\frac{\partial^2 K}{\partial \dot{q}\partial q}$, $\frac{\partial^2 K}{\partial q \partial \dot{q}}$, $\frac{\partial^2 K}{\partial q^2}$ can be recursively computed with Algorithm 4 in $O(n^2)$ time.

Proof. According to Eqs. (2.35), (2.42) and (2.43), we have

$$\frac{\partial K}{\partial \dot{q}_i} = \overline{S}_i^T \Big(\overline{M}_i \overline{v}_i + \sum_{i' \in \mathrm{des}(i)} \overline{M}_{i'} \overline{v}_{i'} \Big), \tag{2.67}$$

$$\frac{\partial K}{\partial q_i} = \dot{\overline{S}}_i^T \Big(\overline{M}_i \overline{v}_i + \sum_{i' \in \mathrm{des}(i)} \overline{M}_{i'} \overline{v}_{i'} \Big). \tag{2.68}$$

Since $\overline{M}_i \overline{v}_i$, $\overline{S}_i$ and $\dot{\overline{S}}_i$ only depend on q_j and $\dot{q}_j$ for $j \in \mathrm{anc}(i) \cup \{i\}$, it is straightforward to show from Eqs. (2.67) and (2.68) that the derivatives $\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j}$, $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j}$, $\frac{\partial^2 K}{\partial q_i \partial \dot{q}_j}$ and $\frac{\partial^2 K}{\partial q_i \partial q_j}$ can be respectively computed as

$$\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j} = \begin{cases} \frac{\partial}{\partial \dot{q}_j} \left(\frac{\partial K}{\partial \dot{q}_i} \right) & j \in \mathrm{anc}(i) \cup \{i\}, \\ \frac{\partial^2 K}{\partial \dot{q}_j \partial \dot{q}_i} & j \in \mathrm{des}(i), \\ 0 & \text{otherwise}, \end{cases} \tag{2.69}$$

$$\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j} = \begin{cases} \frac{\partial}{\partial q_j} \left(\frac{\partial K}{\partial \dot{q}_i} \right) & j \in \mathrm{anc}(i) \cup \{i\}, \\ \frac{\partial^2 K}{\partial q_j \partial \dot{q}_i} & j \in \mathrm{des}(i), \\ 0 & \text{otherwise}, \end{cases} \tag{2.70}$$

$$\frac{\partial^2 K}{\partial q_i \partial \dot{q}_j} = \begin{cases} \frac{\partial}{\partial \dot{q}_j} \left(\frac{\partial K}{\partial q_i} \right) & j \in \mathrm{anc}(i) \cup \{i\}, \\ \frac{\partial^2 K}{\partial \dot{q}_j \partial q_i} & j \in \mathrm{des}(i), \\ 0 & \text{otherwise}, \end{cases} \tag{2.71}$$

$$\frac{\partial^2 K}{\partial q_i \partial q_j} = \begin{cases} \frac{\partial}{\partial q_j}\left(\frac{\partial K}{\partial q_i}\right) & j \in \mathrm{anc}(i) \cup \{i\}, \\ \frac{\partial^2 K}{\partial q_j \partial q_i} & j \in \mathrm{des}(i), \\ 0 & \text{otherwise.} \end{cases} \tag{2.72}$$

Therefore, we only need to consider the derivatives for $j \in \mathrm{anc}(i) \cup \{i\}$, whereas the derivatives for $j \notin \mathrm{anc}(i) \cup \{i\}$ are computed from Eqs. (2.69) to (2.72). In addition, if $j \in \mathrm{anc}(i) \cup \{i\}$, using Eqs. (2.16a), (2.17), (2.18) and (2.20a), we obtain

$$\frac{\partial \overline{M}_i \overline{v}_i}{\partial \dot{q}_j} = \overline{M}_i \overline{S}_j, \tag{2.73}$$

$$\begin{aligned} \frac{\partial \overline{M}_i \overline{v}_i}{\partial q_j} &= -\mathrm{ad}^T_{\overline{S}_j} \overline{M}_i \overline{v}_i - \overline{M}_i \mathrm{ad}_{\overline{S}_j} \overline{v}_i + \overline{M}_i \mathrm{ad}_{\overline{S}_j}(\overline{v}_i - \overline{v}_j) \\ &= \overline{M}_i \dot{\overline{S}}_j - \mathrm{ad}^T_{\overline{S}_j} \overline{M}_i \overline{v}_i \end{aligned} \tag{2.74}$$

$$\frac{\partial \dot{S}_i}{\partial \dot{q}_j} = \mathrm{ad}_{\overline{S}_j} \overline{S}_i, \tag{2.75}$$

$$\frac{\partial \dot{S}_i}{\partial q_j} = \mathrm{ad}_{\overline{v}_i} \mathrm{ad}_{\overline{S}_j} \overline{S}_i + \mathrm{ad}_{\mathrm{ad}_{\overline{S}_j}(\overline{v}_i - \overline{v}_j)} \overline{S}_i. \tag{2.76}$$

For notational clarity, we define $\overline{\mu}_i$, $\overline{\mathcal{M}}_i$, $\overline{\mathcal{M}}^A_i$ and $\overline{\mathcal{M}}^B_i$ as

$$\overline{\mu}_i = \overline{M}_i \overline{v}_i + \sum_{j \in \mathrm{des}(i)} \overline{M}_j \overline{v}_j = \overline{M}_i \overline{v}_i + \sum_{j \in \mathrm{chd}(i)} \overline{\mu}_j, \tag{2.77}$$

$$\overline{\mathcal{M}}_i = \overline{M}_i + \sum_{j \in \mathrm{des}(i)} \overline{M}_j = \overline{M}_i + \sum_{j \in \mathrm{chd}(i)} \overline{\mathcal{M}}_j, \tag{2.78}$$

$$\overline{\mathcal{M}}^A_i = \overline{\mathcal{M}}_i \overline{S}_i, \tag{2.79}$$

$$\overline{\mathcal{M}}^B_i = \overline{\mathcal{M}}_i \dot{\overline{S}}_i - \mathrm{ad}^D_{\overline{\mu}_i} \overline{S}_i \tag{2.80}$$

which will be used in the derivation of $\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j}$, $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j}$, $\frac{\partial^2 K}{\partial q_i \partial \dot{q}_j}$ and $\frac{\partial^2 K}{\partial q_i \partial q_j}$.

1) $\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j}$

If $j \in \text{anc}(i) \cup \{i\}$, from Eqs. (2.67), (2.73), (2.78) and (2.79), it is simple to show that

$$\begin{aligned}
\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j} &= \frac{\partial}{\partial \dot{q}_j}\left(\frac{\partial K}{\partial \dot{q}_i}\right) \\
&= \overline{S}_i^T \Big(\overline{M}_i \overline{S}_j + \sum_{i' \in \text{des}(i)} \overline{M}_{i'} \overline{S}_j \Big) \\
&= \overline{S}_i^T \Big(\overline{M}_i + \sum_{i' \in \text{des}(i)} \overline{M}_{i'} \Big) \overline{S}_j \\
&= \overline{S}_j^T \overline{\mathcal{M}}_i \overline{S}_i \\
&= \overline{S}_j^T \overline{\mathcal{M}}_i^A .
\end{aligned} \tag{2.81}$$

2) $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j}$

If $j \in \text{anc}(i) \cup \{i\}$, using Eqs. (2.13a), (2.67), (2.74), (2.78) and (2.79), we obtain

$$\begin{aligned}
\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j} &= \frac{\partial}{\partial q_j}\left(\frac{\partial K}{\partial \dot{q}_i}\right) \\
&= \sum_{i' \in \text{des}(i) \cup \{i\}} \Big(\overline{S}_i^T \overline{M}_{i'} \dot{\overline{S}}_j - \overline{S}_i^T \,\text{ad}_{\overline{S}_j}^T \overline{M}_{i'} \overline{v}_{i'} + \overline{S}_i^T \,\text{ad}_{\overline{S}_j}^T \overline{M}_{i'} \overline{v}_{i'} \Big) \\
&= \overline{S}_i^T \Big(\overline{M}_i + \sum_{i' \in \text{des}(i)} \overline{M}_{i'} \Big) \dot{\overline{S}}_j \\
&= \dot{\overline{S}}_j^T \overline{\mathcal{M}}_i \overline{S}_i \\
&= \dot{\overline{S}}_j^T \overline{\mathcal{M}}_i^A .
\end{aligned} \tag{2.82}$$

3) $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j}$

If $j \in \mathrm{anc}(i) \cup \{i\}$, using Eqs. (2.68), (2.73), (2.75), (2.77) and (2.78), we obtain

$$\begin{aligned}
\frac{\partial^2 K}{\partial q_i \partial \dot{q}_j} &= \frac{\partial}{\partial \dot{q}_j}\left(\frac{\partial K}{\partial q_i}\right) \\
&= \sum_{i' \in \mathrm{des}(i)\cup\{i\}} \left(\dot{\overline{S}}_i^T \overline{M}_{i'} \overline{S}_j + \overline{S}_i^T \mathrm{ad}_{\overline{S}_j}^T \overline{M}_{i'} \overline{v}_{i'}\right) \\
&= \overline{S}_j^T \Big(\overline{M}_i + \sum_{i' \in \mathrm{des}(i)} \overline{M}_{i'}\Big)\dot{\overline{S}}_i + \Big(\overline{M}_i \overline{v}_i + \sum_{i' \in \mathrm{des}(i)} \overline{M}_{i'} \overline{v}_{i'}\Big)^T \mathrm{ad}_{\overline{S}_j} \overline{S}_i \\
&= \overline{S}_j^T \overline{\mathcal{M}}_i \dot{\overline{S}}_i + \overline{\mu}_i^T \mathrm{ad}_{\overline{S}_j} \overline{S}_i.
\end{aligned}$$

Then simplify the equation above with $\overline{\mu}_i^T \mathrm{ad}_{\overline{S}_j} \overline{S}_i = -\overline{S}_j^T \mathrm{ad}_{\overline{\mu}_i}^D \overline{S}_i$ and Eq. (2.80), the result is

$$\frac{\partial^2 K}{\partial q_i \partial \dot{q}_j} = \overline{S}_j^T \left(\overline{\mathcal{M}}_i \dot{\overline{S}}_i - \mathrm{ad}_{\overline{\mu}_i}^D \overline{S}_i\right) = \overline{S}_j^T \overline{\mathcal{M}}_i^B. \tag{2.83}$$

4) $\frac{\partial^2 K}{\partial q_i \partial q_j}$

If $j \in \mathrm{anc}(i) \cup \{i\}$, using Eqs. (2.18), (2.68), (2.73), (2.74) and (2.76) to (2.78) and $\mathrm{ad}_{\mathrm{ad}_{\overline{v}_i}\overline{S}_j} = \mathrm{ad}_{\overline{v}_i}\mathrm{ad}_{\overline{S}_j} - \mathrm{ad}_{\overline{S}_j}\mathrm{ad}_{\overline{v}_i}$, we obtain

$$\begin{aligned}
\frac{\partial^2 K}{\partial q_i \partial q_j} =& \frac{\partial}{\partial q_j}\left(\frac{\partial K}{\partial q_i}\right) \\
=& \sum_{i' \in \mathrm{des}(i)\cup\{i\}} \Bigg[\left(\overline{M}_{i'} \overline{v}_{i'}\right)^T \Big(\mathrm{ad}_{\overline{v}_i}\mathrm{ad}_{\overline{S}_j}\overline{S}_i - \mathrm{ad}_{\overline{S}_j}\mathrm{ad}_{\overline{v}_i}\overline{S}_i + \\
& \qquad \mathrm{ad}_{\mathrm{ad}_{\overline{S}_j}(\overline{v}_i - \overline{v}_j)}\overline{S}_i\Big) + \dot{\overline{S}}_j^T \overline{M}_{i'} \dot{\overline{S}}_i \Bigg] \\
=& \dot{\overline{S}}_j^T \Big(\overline{M}_i + \sum_{i' \in \mathrm{des}(i)} \overline{M}_{i'}\Big)\dot{\overline{S}}_i + \Big(\overline{M}_i \overline{v}_i + \sum_{i' \in \mathrm{des}(i)} \overline{M}_{i'} \overline{v}_{i'}\Big)^T \mathrm{ad}_{\dot{\overline{S}}_j} \overline{S}_i \\
=& \dot{\overline{S}}_j^T \overline{\mathcal{M}}_i \dot{\overline{S}}_i + \overline{\mu}_i^T \mathrm{ad}_{\dot{\overline{S}}_j} \overline{S}_i.
\end{aligned}$$

Similar to $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j}$, using $\overline{\mu}_i^T \mathrm{ad}_{\dot{\overline{S}}_j} \overline{S}_i = -\dot{\overline{S}}_j^T \mathrm{ad}^D_{\overline{\mu}_i} \overline{S}_i$ and Eq. (2.80), we obtain

$$\frac{\partial^2 K}{\partial q_i \partial q_j} = \dot{\overline{S}}_j^T \left(\overline{\mathcal{M}}_i \dot{\overline{S}}_i - \mathrm{ad}^D_{\overline{\mu}_i} \overline{S}_i \right) = \dot{\overline{S}}_j^T \overline{\mathcal{M}}_i^B . \tag{2.84}$$

Thus far, we have proved that $\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j}$, $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j}$, $\frac{\partial^2 K}{\partial q_i \partial \dot{q}_j}$ and $\frac{\partial^2 K}{\partial q_i \partial q_j}$ can be computed using Eqs. (2.69) to (2.72) and (2.81) to (2.84) with which we further have $\frac{\partial^2 K}{\partial \dot{q}^2}$, $\frac{\partial^2 K}{\partial \dot{q} \partial q}$, $\frac{\partial^2 K}{\partial q \partial \dot{q}}$ and $\frac{\partial^2 K}{\partial q^2}$ computed.

As for the complexity of Algorithm 4, it takes $O(n)$ time to pass the tree representation forward to compute g_i, M_i, $\overline{S}_i$, $\overline{v}_i$, $\dot{\overline{S}}_i$ and another $O(n)$ time to pass the tree representation backward to compute $\overline{\mu}_i$, $\overline{\mathcal{M}}_i$, $\overline{\mathcal{M}}_i^A$ and $\overline{\mathcal{M}}_i^B$. In the backward pass, $\frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j}$, $\frac{\partial^2 K}{\partial \dot{q}_i \partial q_j}$, $\frac{\partial^2 K}{\partial q_i \partial \dot{q}_j}$ and $\frac{\partial^2 K}{\partial q_i \partial q_j}$ are computed for each i using Eqs. (2.69) to (2.72) and (2.81) to (2.84) which totally takes at most $O(n^2)$ time. Therefore, the complexity of Algorithm 4 is $O(n^2)$. This completes the proof. □

2.8.4. Proof of Proposition 2.4

Proposition 2.8.4. If $\vec{g} \in \mathbb{R}^3$ is gravity, then for the gravitational potential energy $V_{\vec{g}}(q)$, $\frac{\partial^2 V_{\vec{g}}}{\partial q^2}$ can be recursively computed with Algorithm 5 in $O(n^2)$ time.

Proof. It is known that the gravitational potential energy $V_{\vec{g}}(q)$ is

$$V_{\vec{g}}(q) = -\sum_{i=1}^{n} m_i \cdot \vec{g}^T p_i. \tag{2.85}$$

in which $m_i \in \mathbb{R}$ is the mass of rigid body i and $p_i \in \mathbb{R}^3$ is the mass center of rigid body i as well as the origin of frame $\{i\}$. In addition, from Eqs. (2.11a) and (2.11b), we have

$$\frac{\partial p_i}{\partial q_j} = \begin{cases} \hat{\overline{s}}_j p_i + \overline{n}_j & j \in \mathrm{anc}(i) \cup \{i\}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.86a}$$

$$\frac{\partial p_j}{\partial q_i} = \begin{cases} \hat{\overline{s}}_i p_j + \overline{n}_i & j \in \mathrm{des}(i) \cup \{i\}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.86b}$$

in which $\overline{s}_i, \overline{n}_i \in \mathbb{R}^3$ and $\overline{S}_i = \begin{bmatrix} \overline{s}_i^T & \overline{n}_i^T \end{bmatrix}^T \in \mathbb{R}^6$ is the spatial Jacobian of joint i. From Eqs. (2.85) and (2.86b), algebraic manipulation gives

$$\frac{\partial V_{\vec{g}}}{\partial q_i} = -\overline{S}_i^T \left(m_i \begin{bmatrix} \hat{p}_i \vec{g} \\ \vec{g} \end{bmatrix} + \sum_{i' \in \mathrm{des}(i)} m_{i'} \begin{bmatrix} \hat{p}_{i'} \vec{g} \\ \vec{g} \end{bmatrix} \right). \tag{2.87}$$

Moreover, observe that $\overline{S}_i$ and p_i only depends on q_j for $j \in \mathrm{anc}(i) \cup \{i\}$, we obtain from Eq. (2.87) that

$$\frac{\partial^2 V_{\vec{g}}}{\partial q_i \partial q_j} = \begin{cases} \frac{\partial}{\partial q_j} \left(\frac{\partial V_{\vec{g}}}{\partial q_i} \right) & j \in \mathrm{anc}(i) \cup \{i\}, \\ \frac{\partial^2 V_{\vec{g}}}{\partial q_j \partial q_i} & j \in \mathrm{des}(i), \\ 0 & \text{otherwise}, \end{cases} \tag{2.88}$$

which means that only $\frac{\partial^2 V_{\vec{g}}}{\partial q_i \partial q_j}$ for $j \in \mathrm{anc}(i) \cup \{i\}$ needs to be explicitly computed. If $j \in \mathrm{anc}(i) \cup \{i\}$, using Eqs. (2.13a), (2.86a) and (2.87) as well as the equality $\hat{a}b = -\hat{b}a$ for any $a, b \in \mathbb{R}^3$, we obtain

$$\begin{aligned} \frac{\partial^2 V_{\vec{g}}}{\partial q_i \partial q_j} &= \frac{\partial}{\partial q_j} \left(\frac{\partial V_{\vec{g}}}{\partial q_i} \right) \\ &= \sum_{i' \in \mathrm{des}(i) \cup \{i\}} m_{i'} \left[\overline{s}_i^T \left(\hat{\vec{g}} \hat{\overline{s}}_j p_{i'} + \hat{\overline{s}}_j \hat{p}_{i'} \vec{g} \right) - \overline{n}_i^T \hat{\vec{g}} \overline{s}_j \right]. \end{aligned}$$

In addition, since $\hat{p}_{i'}\hat{\vec{g}}\overline{s}_j = -\hat{\vec{g}}\hat{\overline{s}}_j p_{i'} - \hat{\overline{s}}_j \hat{p}_{i'}\vec{g}$ and $\hat{a}^T = -\hat{a}$ for any $a \in \mathbb{R}^3$, the equation above is equivalent to

$$\frac{\partial^2 V_{\vec{g}}}{\partial q_i \partial q_j} = \overline{s}_j^T \hat{\vec{g}} \left[\big(m_i + \sum_{i' \in \mathrm{des}(i)} m_{i'}\big)\overline{n}_i - \big(m_i \hat{p}_i + \sum_{i' \in \mathrm{des}(i)} m_{i'}\hat{p}_{i'}\big)\overline{s}_i \right] \tag{2.89}$$

If we define

$$\overline{\sigma}_{m_i} = m_i + \sum_{j \in \mathrm{des}(i)} m_j = m_i + \sum_{j \in \mathrm{chd}(i)} \overline{\sigma}_{m_j},$$

$$\overline{\sigma}_{p_i} = m_i p_i + \sum_{j \in \mathrm{des}(i)} m_j p_j = m_i p_i + \sum_{j \in \mathrm{chd}(i)} \overline{\sigma}_{p_j},$$

$$\overline{\sigma}_i^A = \hat{\vec{g}} \left(\overline{\sigma}_{m_i} \cdot \overline{n}_i - \hat{\overline{\sigma}}_{p_i} \cdot \overline{s}_i \right),$$

then Eq. (2.89) is further simplified to

$$\frac{\partial^2 V_{\vec{g}}}{\partial q_i \partial q_j} = \overline{s}_j^T \hat{\vec{g}} \left(\overline{\sigma}_{m_i} \overline{n}_i - \hat{\overline{\sigma}}_{p_i} \overline{s}_i \right) = \overline{s}_j^T \overline{\sigma}_i^A. \tag{2.90}$$

As a result, $\frac{\partial^2 V_{\vec{g}}}{\partial q^2}$ can be computed from Eqs. (2.88) and (2.90).

The $O(n^2)$ complexity of Algorithm 5 is as follows: the forward pass to compute g_i and $\overline{S}_i$ and the backward pass to compute $\overline{\sigma}_{m_i}$, $\overline{\sigma}_{p_i}$ and $\overline{\sigma}_i^A$ take $O(n)$ time, respectively; and the computation of $\frac{\partial^2 V_{\vec{g}}}{\partial q_i \partial q_j} = \frac{\partial^2 V_{\vec{g}}}{\partial q_j \partial q_i} = \overline{s}_j^T \overline{\sigma}_i^A$ totally takes $O(n^2)$ time. Therefore, it can be concluded that Algorithm 5 has $O(n^2)$ complexity. This completes the proof. $\square$

CHAPTER 3

Efficient and Certifiably Correct Planar Graph-Based SLAM Using the Complex Number Representation

This chapter considers the problem of planar graph-based simultaneous localization and mapping (SLAM) that involves both poses of the autonomous agent and positions of observed landmarks. We present CPL-SLAM, an efficient and certifiably correct algorithm to solve planar graph-based SLAM using the complex number representation. We formulate and simplify planar graph-based SLAM as the maximum likelihood estimation (MLE) on the product of unit complex numbers, and relax this nonconvex quadratic complex optimization problem to convex complex semidefinite programming (SDP). Furthermore, we simplify the corresponding complex semidefinite programming to Riemannian staircase optimization (RSO) on the complex oblique manifold that can be solved with the Riemannian trust region (RTR) method. In addition, we prove that the SDP relaxation and RSO simplification are tight as long as the noise magnitude is below a certain threshold. The efficacy of this work is validated through applications of CPL-SLAM and comparisons with existing state-of-the-art methods on planar graph-based SLAM, which indicates that our proposed algorithm is capable of solving planar graph-based SLAM certifiably, and is more efficient in numerical computation and more robust to measurement noise than existing state-of-the-art methods.

3.1. Introduction

Simultaneous localization and mapping (SLAM) estimates poses of an autonomous agent and positions of observed landmarks from noisy measurements [**44**–**46**]. For an autonomous agent, the ability to construct a map of the environment and concurrently estimate its location within the map is essential to navigation and exploration in unknown scenarios, such as autonomous driving [**47**], disaster response [**48**], underwater exploration [**49**], precision agriculture [**50**], floor plan building [**51**], virtual and augmented reality [**52**], to name a few. An intuitive way to formulate SLAM problems is to use a graph whose vertices are associated with either poses of the autonomous agent or positions of observed landmarks and whose edges are associated with the available measurements [**17**]. In graph-based SLAM, the estimation problem is usually addressed as a difficult nonconvex optimization problem that involves up to thousands of variables and constraints, and the procedure of solving the optimization problem greatly affects the overall performance of estimation. Even though a number of optimization methods have been developed [**17**, **44**, **53**], it is generally NP-hard to solve a nonconvex optimization problem globally [**53**], and it is common to get stuck at local minima in solving graph-based SLAM, which results in bad estimates.

In robotics, most graph-based SLAM techniques rely on local search methods for nonlinear optimization to estimate poses of the autonomous agent and positions of observed landmarks. Lu and Milios [**54**] formulate SLAM as pose graph optimization (PGO) and use iterative nonlinear optimization methods to solve PGO. Duckett and Frese et al. [**55**, **56**] exploit the sparsity of graph-based SLAM and propose relaxations for the resulting nonlinear optimization problem. Olson et al. [**57**] propose a stochastic gradient

descent on an alternative state space representation of graph-based SLAM that has good stability and scalability. Grisetti et al. [**58**] extend Olson's work by presenting a novel tree parametrization that improves the convergence of stochastic gradient descent. Fan and Murphey [**59**] propose an accelerated proximal method for pose graph optimization. Dellaert and Kaess et al. [**60**–**63**] propose incremental smoothing algorithms that enable online updates of large-scale graph-based SLAM with nonlinear optimization. Huang and Wang et al. [**64**, **65**] study the least-square structure of graph-based SLAM and indicate the possibility of reducing the nonlinearity and nonconvexity of SLAM. Kümmerle et al. [**66**] propose g^2o framework that solves graph-based SLAM using Gauss-Newton method. Carlone et al. [**67**, **68**] propose approximations for planar pose graph optimization that reduce the risks of getting stuck at local minima. Khosoussi et al. [**69**] exploit the separable structure of SLAM problems using variable projection and propose algorithms to improve the efficiency of Gauss-Newton methods. However, all of the aforementioned nonlinear optimization techniques are local search methods, and as a result, there are no guarantees of the correctness for the resulting solutions.

To address the issues of local minima in nonlinear optimization, several efforts have been made to relax graph-based SLAM as convex optimization problems. Liu et al. [**70**] propose a suboptimal convex relaxation to solve SLAM problems. Carlone et al. [**71**–**73**] propose a tight semidefinite relaxation and analyze its optimality using Lagrangian duality. Briales et al. [**74**] present a fast method for the optimality verification of 3D PGO based on [**73**]. A further breakthrough of the semidefinite relaxation of PGO is made in [**7**] that results in a fast and certifiable algorithm for pose graph optimization. Rosen et al. propose SE-Sync to solve the semidefinite relaxation of PGO using Riemannian

staircase optimization on the Stiefel manifold that is orders of magnitude faster than interior point methods [**7**]. Furthermore, it is shown in [**7**] that the semidefinite relaxation of PGO is tight as long as the magnitude of the measurement noise is below a certain threshold. Similar to SE-Sync, Briales et al. [**75**] propose Cartan-Sync that uses the Cartan motion group and introduce a novel preconditioner to accelerate the algorithm. Mangelson et al. [**76**] formulate planar pose graph and landmark SLAM using sparse-bounded sum of squares programming that is guaranteed to find the globally optimal solution regardless of the noise level.

In fields other than robotics, problems such as angular and rotation synchronization that share a similar mathematical formulation with graph-based SLAM have been extensively studied. Singer et al. [**77**, **78**] propose semidefinite relaxations to solve angular and rotation synchronization by finding the eigenvectors that correspond to the greatest eigenvalues. Bandeira et al. [**79**] prove the tightness of the semidefinite relaxation of angular synchronization and show that the Riemannian staircase optimization is significantly more scalable to solve the resulting problem. Boumal [**80**] proposes the generalized power method that can recover the globally optimal solution to angular synchronization. Eriksson et al. [**81**] explore the role of strong duality in rotation averaging, which has important applications in computer vision.

In applied mathematics, it is common to use unit complex numbers in synchronization problems over $SO(2)$ [**79**, **80**]. In robotics, it is not new to use the complex number representation in planar robot localization and mapping problems, either. Betke et al. [**82**] use the complex number to represent positions of landmarks to localize a mobile robot with bearing measurements. Carlone et al. [**72**] use the complex number representation of

$SO(2)$ and $SE(2)$ to verify the optimality of planar PGO and the tightness of semidefinite relaxations, and the analysis is much clearer and simpler than that using the matrix representation, and to our knowledge, this is the first implementation of the complex number representation in planar PGO.

In general, a certifiably correct algorithm for an optimization problem not only finds a solution to the problem, but also is capable of certifying the global optimality of the resulting solution [**83**]. For many estimation problems, it is usually intractable to attain a globally optimal solution and we have to either solve these problems using local search methods, or relax them to a more reasonable formulation. As a result, the certifiable correctness of the algorithm is important for estimation problems in which globally optimal solutions are preferred. Even though a number of optimization methods are proposed to planar graph-based SLAM, to our knowledge, only [**7**, **71**–**76**] are certifiably correct.

In this chapter, we consider the problem of planar graph-based SLAM that involves both poses of the autonomous agent and positions of observed landmarks. We present CPL-SLAM, which means the **C**om**PL**ex number **S**imultaneous **L**ocalization **A**nd **M**apping, an efficient and certifiably correct algorithm to solve planar graph-based SLAM using the complex number representation.

This chapter is built upon the works of [**7**, **72**, **79**] that use the complex number representation, the semidefinite relaxation and the Riemannian staircase optimization [**84**] to efficiently and certifiably correctly solve large-scale estimation problems. In [**72**], Carlone et al. were first to formulate planar PGO using the complex number representation of $SE(2)$, in which the optimality and tightness of the semidefinite relaxation are studied; in CPL-SLAM, we use the same representation of $SE(2)$ as the one in [**72**]. In [**7**],

Rosen et al. analyze the optimality and tightness of the semidefinite relaxation of PGO and introduce the Riemannian staircase optimization to solve the semidefinite relaxation, which, though using the matrix representation, motivates this chapter. In **[79]**, Bandeira et al. prove the tightness of semidefinite relaxation of phase synchronization on $SO(2)$ using the complex number representation, which is helpful to our theoretical analysis of CPL-SLAM.

In graph-based SLAM, poses are special Euclidean groups $SE(d)$ **[7, 17, 63, 66, 75]**, which are isomorphic to a semidirect product of real space $\mathbb{R}^d$ and special orthogonal groups $SO(d)$ **[85]**. In general, it is possible to identify $SE(d)$ as a pair $(t,\, R)$, in which the translation is represented as a real vector t in $\mathbb{R}^d$ and the rotation is represented as a matrix R in $SO(d)$, and such an identification results in the matrix representation of $SE(d)$ that is commonly used in robotics. However, for planar graph-based SLAM, the matrix representation of $SO(2)$ is redundant, which needs four real numbers, whereas a unit complex number that can be represented with two real numbers is sufficient to capture the topological and geometric structures of $SO(2)$ **[72, 86]**. Furthermore, as is later shown in this chapter, the complex number representation of $SO(2)$ and $SE(2)$ brings significant analytical and computational benefits, and renders the resulting CPL-SLAM algorithm a lot more efficient in numerical computation and much more robust to measurement noise. As a result, the CPL-SLAM outperforms existing methods of planar graph-based SLAM in terms of both numerical scalability and theoretical guarantees.

In contrast to the state-of-the-art local search methods in **[54–68]**, CPL-SLAM is a lot faster and capable of certifying the correctness of the solutions. As for **[7, 72, 76]** that also seek to use convex relaxation to certifiably correctly solve graph-based SLAM, CPL-SLAM

has better scalability, and more explicitly, CPL-SLAM is expected to be several orders of magnitude faster than [**72**, **76**] and several times faster than [**7**]. Moreover, [**7**, **72**] are designed for pose-only problems, whereas CPL-SLAM extends the works of [**7**, **72**] by accepting pose-landmark measurements. Even though [**72**] uses complex semidefinite relaxation to verify the optimality and tightness of planar pose graph optimization, we present stronger, more complete and more concise theoretical results and more scalable algorithms to solve the complex semidefinite relaxation. Furthermore, the conciseness of the complex number representation makes the semidefinite relaxation in CPL-SLAM much tighter than that in [**7**] using the matrix representation, and thus, CPL-SLAM has greater robustness to measurement noise.

This chapter is based on the preliminary results of [**87**, **88**] where we use the complex number representation to solve planar graph-based SLAM with landmarks. Similar to [**7**, **75**, **87**] for pose graph optimization, CPL-SLAM is a certifiable algorithm that is guaranteed to attain the globally optimal solution to planar graph-based SLAM with landmarks as long as the magnitude of measurement noise is below a certain threshold. Furthermore, even though it is not new to involve landmarks in graph-based SLAM [**17**, **58**, **89**], we propose a novel preconditioner making better use of translation and landmark information in planar graph-based SLAM. As a result, the performance of the truncated conjugate gradient method is improved. In addition, we also provide the proofs of lemmas and propositions, extensive experimental results on numerous datasets and much more detailed discussions.

In summary, the contributions of this chapter are the following:

(1) We formulate planar graph-based SLAM with poses of the autonomous agent and positions of observable landmarks using the complex number representation and simplify the resulting estimation problem as an optimization problem on the product of unit complex numbers.

(2) We relax the nonconvex optimization problem as complex semidefinite programming and prove that the complex semidefinite relaxation is tight as long as the magnitude of measurement noise is below a certain threshold.

(3) We recast the complex semidefinite programming as a series of rank-restricted complex semidefinite programming on complex oblique manifolds that can be efficiently solved with the Riemannian staircase optimization [**84**], and it is almost guaranteed to retrieve the true solution to the complex semidefinite programming if the rank of the Riemannian staircase optimization is appropriately selected.

(4) The resulting CPL-SLAM algorithm is certifiably correct, and more importantly, a lot faster in numerical computation and much more robust to measurement noise than existing state-of-the-art methods [**7**, **54**–**58**, **60**–**68**, **72**, **89**].

The rest of this chapter is organized as follows. Section 3.2 introduces notations that are used throughout this chapter. Sections 3.3 and 3.4 review the complex number representation of $SO(2)$ and $SE(2)$ and the complex oblique manifold [**90**]. Section 3.5 formulates planar graph-based SLAM with poses of the autonomous agent and positions of observable landmarks using the complex representation and Section 3.6 relaxes planar graph-based SLAM to complex semidefinite programming. Section 3.7 presents the CPL-SLAM algorithm to solve planar graph-based SLAM. Section 3.8 presents and discusses comparisons of CPL-SLAM with existing methods [**7**, **63**, **89**] on a series of simulated

Tree and City datasets and a suite of large 2D simulated and real-world SLAM benchmark datasets. The conclusions are made in Section 3.9. The proofs of the lemmas and propositions are presented in Section 3.10.

3.2. Notation

$\mathbb{R}$ and $\mathbb{C}$ denote the sets of real and complex numbers, respectively; $\mathbb{R}^{m\times n}$ and $\mathbb{C}^{m\times n}$ denote the sets of $m \times n$ real and complex matrices, respectively; $\mathbb{R}^n$ and $\mathbb{C}^n$ denote the sets of $n \times 1$ real and complex vectors, respectively. $\mathbb{C}_1$ and $\mathbb{C}_1^n$ denote the sets of unit complex numbers and $n\times 1$ vectors over unit complex numbers, respectively. $\mathbb{P}$ denotes the group of $(\mathbb{C}, +) \rtimes (\mathbb{C}_1, \cdot)$ in which "$\rtimes$" denotes the semidirect product of groups **[85]** under complex number addition "+" and multiplication "$\cdot$". $\mathbb{S}^n$ and $\mathbb{H}^n$ denote the sets of $n \times n$ real symmetric matrices and complex Hermitian matrices, respectively. The notation "$\mathbf{i}$" is reserved for the imaginary unit of complex numbers. The notation $|\cdot|$ denotes the absolute value of real and complex numbers, and the notation $\overline{(\cdot)}$ denote the conjugate of complex numbers. The superscripts $(\cdot)^\top$ and $(\cdot)^H$ denote the transpose and conjugate transpose of a matrix, respectively. For a complex matrix W, $[W]_{ij}$ denotes its (i, j)-th entry; the notations $\Re(W)$ and $\Im(W)$ denote real matrices such that $W = \Re(W) + \Im(W)\mathbf{i}$; $W \succcurlyeq 0$ means that W is Hermitian and positive semidefinite; trace(W) denotes the trace of W; diag(W) extracts the diagonal of W into a vector and ddiag(W) sets all off-diagonal entries of W to zero; the notations $\|W\|_F$ and $\|W\|_2$ denote the Frobenius norm and the induced-2 norm, respectively. The notation $\langle \cdot, \cdot \rangle$ denotes the real inner product of matrices. For a vector v, the notation $[v]_i$ denotes its i-th entry; $\|v\|^2 = \|v\|_2^2 = \sqrt{\sum_i |[v]_i|^2} = \sqrt{v^H v}$; the notation diag$(v)$ denotes the diagonal matrix with $\big[\text{diag}(v)\big]_{ii} = v_i$. The notation $\mathbf{1} \in \mathbb{C}^n$

denotes the vector of all-ones. The notation $\mathbf{0} \in \mathbb{C}^n$ denotes the vector of all-zeros. The notation $\mathbf{I} \in \mathbb{C}^{n \times n}$ denotes the identity matrix. The notation $\mathbf{O} \in \mathbb{C}^{n \times n}$ denotes the zero matrix. For a hidden parameter x whose value we wish to infer, the notations $\underline{x}$, $\tilde{x}$ and $\hat{x}$ denote the true value of x, a noisy observation of x and an estimate of x, respectively.

3.3. The Complex Number Representation of $SO(2)$ and $SE(2)$

In this section, we give a brief review of $SO(2)$ and $SE(2)$, and show that $SO(2)$ and $SE(2)$ can be represented using complex numbers. It should be noted that the complex number representation used in this chapter, though presented in a different way, is in fact equivalent to that in **[72]**.

It is known that the set of unit complex numbers

$$\mathbb{C}_1 \triangleq \{a_1 + a_2\mathbf{i} \in \mathbb{C} | a_1^2 + a_2^2 = 1\}$$

forms a group under complex number multiplication "$\cdot$" for which the identity is 1 and the inverse is the conjugate, i.e., for $z, z' \in \mathbb{C}_1$, we obtain **[86]**

$$z \cdot z' \in \mathbb{C}_1, \quad 1 \cdot z = z \cdot 1 = z, \quad z \cdot \overline{z} = \overline{z} \cdot z = 1.$$

In addition, the group of unit complex numbers $(\mathbb{C}_1, \cdot)$ is diffeomorphic and isomorphic to the matrix Lie group $SO(2)$:

$$\begin{aligned} SO(2) &\triangleq \{ \begin{bmatrix} a_1 & -a_2 \\ a_2 & a_1 \end{bmatrix} \in \mathbb{R}^{2\times 2} | a_1^2 + a_2^2 = 1\} \\ &\triangleq \{R \in \mathbb{R}^{2\times 2} | R^\top R = \mathbf{I}, \det(R) = 1\} \end{aligned}$$

under matrix multiplication. As a result, $SO(2)$ can be represented using unit complex numbers $\mathbb{C}_1$. More explicitly, if $R \in SO(2)$ is

$$R = \begin{bmatrix} a_1 & -a_2 \\ a_2 & a_1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{3.1}$$

the corresponding unit complex number representation $z \in \mathbb{C}_1$ is

$$z = a_1 + a_2\mathbf{i} = e^{\mathbf{i}\theta} = \cos\theta + \sin\theta\mathbf{i} \tag{3.2}$$

in which $e^{\mathbf{i}\theta} = \cos\theta + \sin\theta\mathbf{i}$. Furthermore, if $b' = \begin{bmatrix} b'_1 & b'_2 \end{bmatrix}^\top \in \mathbb{R}^2$ is rotated by $R \in SO(2)$ in Eq. (3.1) from $b = \begin{bmatrix} b_1 & b_2 \end{bmatrix}^\top \in \mathbb{R}^2$, i.e.,

$$b'_1 = a_1 b_1 - a_2 b_2 = b_1 \cos\theta - b_2 \sin\theta,$$

$$b'_2 = a_1 b_2 + a_2 b_1 = b_2 \cos\theta + b_1 \sin\theta,$$

we obtain

$$s' = z \cdot s = \underbrace{a_1 b_1 - a_2 b_2}_{b'_1} + (\underbrace{a_1 b_2 + a_2 b_1}_{b'_2})\mathbf{i}, \tag{3.3a}$$

or equivalently,

$$\begin{aligned} s' &= z \cdot s = e^{\mathbf{i}\theta} \cdot s \\ &= \underbrace{b_1 \cos\theta - b_2 \sin\theta}_{b'_1} + (\underbrace{b_2 \cos\theta + b_1 \sin\theta}_{b'_2})\mathbf{i}, \end{aligned} \tag{3.3b}$$

in which z is a unit complex number as that given in Eq. (3.2), and

$$s = b_1 + b_2\mathbf{i} \quad \text{and} \quad s' = b'_1 + b'_2\mathbf{i} \tag{3.4}$$

are the complex number representation of b and b', respectively. As a result, rotating a vector can also be described using the complex number representation.

In general, the special Euclidean group $SE(2)$ is the matrix Lie group

$$SE(2) \triangleq \{ \begin{bmatrix} R & t \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} | R \in SO(2),\, t \in \mathbb{R}^2 \}, \tag{3.5}$$

whose group multiplication is matrix multiplication. In terms of group theory, $SE(2)$ is also represented as the semidirect product of $(\mathbb{R}^2, +)$ and $SO(2)$:

$$SE(2) \triangleq (\mathbb{R}^2, +) \rtimes SO(2),$$

in which "$\rtimes$" denotes the semidirect product of groups under vector addition and matrix multiplication [**85**] and whose group multiplication "$\circ$" using the matrix representation of Eq. (3.5) is defined to be

$$g \circ g' = (Rt' + t, RR'), \tag{3.6}$$

in which $g = (t, R)$, $g' = (t', R') \in SE(2)$.[1] Following the complex number representation of $SO(2)$ and $\mathbb{R}^2$, the representation of $SE(2)$ as Eq. (3.5) is diffeomorphic and isomorphic to the semidirect product of $(\mathbb{C}, +)$ and $(\mathbb{C}_1, \cdot)$:

$$\mathbb{P} \triangleq (\mathbb{C}, +) \rtimes (\mathbb{C}_1, \cdot),$$

[1] In group theory, the definition of the semidirect product relies on the choice of the group multiplication rule.

whose group multiplication "$\odot$" is defined to be

$$\rho \odot \rho' = (z \cdot c' + c, z \cdot z') \in \mathbb{P}, \tag{3.7}$$

in which $\rho = (c, z)$, $\rho' = (c', z') \in \mathbb{P}$. In Eq. (3.7), z, $z' \in \mathbb{C}_1$ and c, $c' \in \mathbb{C}$ are the complex number representation of R, $R' \in SO(2)$ and t, $t' \in \mathbb{R}^2$, respectively, which follow the same representation as that in Eqs. (3.2) and (3.4). It is obvious from Eqs. (3.2) and (3.3) that the group multiplication of $\rho \odot \rho'$ in Eq. (3.7) is equivalent to that of $g \circ g'$ in Eq. (3.6). Furthermore, the identity of $\mathbb{P}$ is $(0, 1) \in \mathbb{P}$ and the inverse of $\rho = (c, z) \in \mathbb{P}$ is

$$\rho^{-1} = (-\overline{z} \cdot c, \overline{z}) \in \mathbb{P}. \tag{3.8}$$

As a result, instead of using the matrix representation, we represent $SE(2)$ with a 2-tuple of complex numbers. In addition, if $b' \in \mathbb{R}^2$ is transformed by $g \in SE(2)$ from $b \in \mathbb{R}^2$, we obtain

$$s' = z \cdot s + c,$$

in which $\rho = (c,\ z) \in \mathbb{P}$ is the complex number representation of $g \in SE(2)$, and s and s' are the complex number representation of b and b', respectively.

For notational convenience, in the rest of paper, we will omit the complex number multiplication "$\cdot$" if there is no ambiguity.

In terms of the computation of group multiplication and transformation only, the complex number representation of $SO(2)$ and $SE(2)$ has the same complexity as the matrix representation. In spite of this, as shown in the following sections, the complex number representation greatly simplifies the analysis for planar graph-based SLAM, and most importantly, the semidefinite relaxation and Riemannian optimization of planar graph-based

SLAM using the complex number representation is simpler for problem formulation, more efficient in numerical computation and more robust to measurement noise than that using the matrix representation in [**7**].

In this chapter, we will use the complex number representation of $SO(2)$ and $SE(2)$ to formulate and solve planar graph-based SLAM.

3.4. The Complex Oblique Manifold

In this section, we give a brief review of the Riemannian geometric concepts and operators of the complex oblique manifold [**91**] that are used in this chapter. A detailed introduction to Riemannian geometry and optimization can be found in [**90**].

In Riemannian geometry, the complex oblique manifold

$$\mathrm{OB}(r,n) \triangleq \{Y \in \mathbb{C}^{n\times r} | \mathrm{ddiag}(YY^H) = \mathbf{I}\}$$

is a smooth and compact complex matrix manifold, whose tangent space at $Y \in \mathrm{OB}(r,n)$ is

$$T_Y\mathrm{OB}(r,n) \triangleq \{U \in \mathbb{C}^{n\times r} | \Re\{\mathrm{ddiag}(UY^H)\} = \mathbf{O}\}.$$

For any $U \in \mathbb{C}^{n\times r}$, we define the projection operator $\mathrm{proj}_Y : \mathbb{C}^{n\times r} \to T_Y\mathrm{OB}(r,n)$ to be

$$\mathrm{proj}_Y U \triangleq U - \Re\{\mathrm{ddiag}(UY^H)\}Y.$$

7 For a smooth function $F : \mathrm{OB}(r,n) \to \mathbb{R}$, it is by definition that the Riemannian gradient for $F(Y)$ is

$$\mathrm{grad}\, F(Y) \triangleq \nabla F(Y) - \Re\{\mathrm{ddiag}(\nabla F(Y)Y^H)\}Y.$$

Since we have assumed that the complex oblique manifold is a Riemannian submanifold of Euclidean space, then for any tangent vector $\dot{Y} \in T_Y \mathrm{OB}(r, n)$, the Riemannian Hessian for $F(Y)$ can be computed as

$$\operatorname{Hess} F(Y)[\dot{Y}] \triangleq \operatorname{proj}_Y \operatorname{Dgrad} F(Y)[\dot{Y}],$$

in which $\mathrm{D}\operatorname{grad} F(z)[\dot{z}]$ is the direction derivative of $\operatorname{grad} F(z)$ along direction $\dot{z}$.

3.5. Problem Formulation and Simplification

In this section, we formulate planar graph-based SLAM as maximum likelihood estimation, and further simplify it to complex quadratic programming on the product of unit complex numbers.

3.5.1. Problem Formulation

Planar graph-based SLAM consists of estimating n unknown poses $g_1, g_2, \cdots, g_n \in SE(2)$, in which $g_{(\cdot)} = (t_{(\cdot)}, R_{(\cdot)})$ with $t_{(\cdot)} \in \mathbb{R}^2$ and $R_{(\cdot)} \in SO(2)$, and n' landmark positions $l_1, l_2, \cdots, l_{n'} \in \mathbb{R}^2$ given m noisy pose-pose measurements $\tilde{g}_{ij} \in SE(2)$ of

$$g_{ij} \triangleq g_i^{-1} g_j \in SE(2)$$

and m' noisy pose-landmark measurements $\tilde{l}_{ij} \in \mathbb{R}^2$ of

$$l_{ij} \triangleq R_i^\top (l_j - t_i) \in \mathbb{R}^2. \tag{3.9}$$

From Section 3.3, the problem is equivalent to estimating n 2-tuples of complex numbers $\rho_1, \rho_2, \cdots, \rho_n \in \mathbb{P}$, in which $\rho_{(\cdot)} = (c_{(\cdot)}, z_{(\cdot)}) \in \mathbb{P}$ with $c_{(\cdot)} \in \mathbb{C}$ and $z_{(\cdot)} \in \mathbb{C}_1$, and n'

complex numbers $s_1, s_2, \cdots, s_{n'} \in \mathbb{C}$ given m noisy pose-pose measurements $\tilde{\rho}_{ij} \in \mathbb{P}$ of

$$\rho_{ij} \triangleq \rho_i^{-1} \odot \rho_j \in \mathbb{P}$$

and m' noisy pose-landmark measurements $\tilde{s}_{ij} \in \mathbb{C}$ of

$$s_{ij} \triangleq \overline{z}_i(s_j - c_i) \in \mathbb{C}.$$

The unknown n poses and n' landmark positions and the noisy relative measurements can be described with a directed graph $\overrightarrow{G} = (\mathcal{V} \cup \mathcal{V}', \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}'})$ in which $i \in \mathcal{V} \triangleq \{1, \cdots, n\}$ is associated with g_i or ρ_i, and $i \in \mathcal{V}' = \{1, \cdots, n'\}$ is associated with l_i or s_i, and $(i,j) \in \overrightarrow{\mathcal{E}} \subset \mathcal{V} \times \mathcal{V}$ if and only if the pose-pose measurement $\tilde{g}_{ij}$ or $\tilde{\rho}_{ij}$ exists, and $(i,j) \in \overrightarrow{\mathcal{E}'} \subset \mathcal{V} \times \mathcal{V}'$ if and only if the pose-landmark measurement $\tilde{l}_{ij}$ or $\tilde{s}_{ij}$ exists. If the orientation of edges in $\overrightarrow{\mathcal{E}}$ and $\overrightarrow{\mathcal{E}'}$ are ignored, we obtain the undirected graph of $\overrightarrow{G}$ that is denoted as $G = (\mathcal{V} \cup \mathcal{V}', \mathcal{E} \cup \mathcal{E}')$. In the rest of this chapter, we assume that $\overrightarrow{G}$ is weakly connected and G is (equivalently) connected. In addition, we assume that the noisy relative measurements $\tilde{\rho}_{ij} = (\tilde{c}_{ij}, \tilde{z}_{ij})$ and $\tilde{s}_{ij}$ are random variables that satisfy

$$\tilde{c}_{ij} = \underline{c}_{ij} + c_{ij}^{\epsilon} \qquad c_{ij}^{\epsilon} \sim N(0, \tau_{ij}^{-1}), \tag{3.10a}$$

$$\tilde{z}_{ij} = \underline{z}_{ij} z_{ij}^{\epsilon} \qquad \tilde{z}_{ij}^{\epsilon} \sim \mathrm{vMF}(1, \kappa_{ij}), \tag{3.10b}$$

$$\tilde{s}_{ij} = \underline{s}_{ij} + s_{ij}^{\epsilon} \qquad s_{ij}^{\epsilon} \sim N(0, \nu_{ij}^{-1}), \tag{3.10c}$$

for all $(i,j) \in \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}'}$. In Eq. (3.10), $\underline{\rho}_{ij} = (\underline{c}_{ij}, \underline{z}_{ij})$ and $\underline{s}_{ij}$ are the true (latent) values of ρ_{ij} and s_{ij}, respectively, $N(\mu, \Sigma)$ denotes the complex normal distribution with mean $\mu \in \mathbb{C}$ and covariance $\Sigma \succcurlyeq 0$, and $\mathrm{vMF}(z_0, \kappa)$ denotes the von Mises-Fisher distribution on

$\mathbb{C}_1$ with mode $z_0 \in \mathbb{C}_1$, concentration number $\kappa \geq 0$ and the probability density function of $\text{vMF}(z_0, \kappa)$ is **[92]**

$$f(z; z_0, \kappa) = \frac{1}{c_d(\kappa)} \exp\left(\kappa(\overline{z}_0 z + z_0 \overline{z})\right),$$

in which $c_d(\kappa)$ is a function of κ.

If $\tilde{c}_{ij}$, $\tilde{z}_{ij}$ and $\tilde{s}_{ij}$ are independent from each other, from Eqs. (3.3), (3.7) and (3.8), a straightforward algebraic manipulation indicates that the maximum likelihood estimation (MLE) is a least square problem as follows

$$\text{(MLE)} \quad \min_{\substack{s_i \in \mathbb{C}, \\ c_i \in \mathbb{C}^n,\, z_i \in \mathbb{C}_1^n}} \sum_{(i,j) \in \overrightarrow{\mathcal{E}}} \left[\kappa_{ij} |z_i \tilde{z}_{ij} - z_j|^2 + \tau_{ij} |c_j - c_i - z_i \tilde{c}_{ij}|^2\right] + \sum_{(i,j) \in \overrightarrow{\mathcal{E}'}} \nu_{ij} |s_j - c_i - z_i \tilde{s}_{ij}|^2,$$

in which κ_{ij}, τ_{ij} and ν_{ij} are as given in Eqs. (3.10a) to (3.10c). From Eqs. (3.1) and (3.2), it should be noted that

$$|z_i \tilde{z}_{ij} - z_j|^2 = \frac{1}{2} \|R_i \widetilde{R}_{ij} - R_j\|_F^2,$$

and it is also trivial to show that

$$|c_j - c_i - z_i \tilde{c}_{ij}|^2 = \|t_j - t_i - R_i \tilde{t}_{ij}\|_F^2,$$

$$|s_j - c_i - z_i \tilde{s}_{ij}|^2 = \|l_j - t_i - R_i \tilde{l}_{ij}\|_F^2.$$

As a result, (MLE) is equivalent to

$$\text{(SE-MLE)} \quad \min_{\substack{R_i\in SO(2),\\ p_i,\, l_i\in\mathbb{R}^2}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}} \Big[\frac{\kappa_{ij}}{2}\|R_i\widetilde{R}_{ij}-R_j\|_F^2+\tau_{ij}\|t_j-t_i- R_i\tilde{t}_{ij}\|_F^2\Big] + \sum_{(i,j)\in\overrightarrow{\mathcal{E}'}} \nu_{ij}\|l_j-t_i-R_il_{ij}\|_F^2.$$

Even though there are landmarks present in (MLE) and (SE-MLE), this does not create a significant distinction from [**7**] in terms of problem formulation. As a matter of fact, if there are no landmarks, i.e., $\mathcal{V}'=\emptyset$ and $\overrightarrow{\mathcal{E}'}=\emptyset$, (SE-MLE) is almost the same as the formulation of pose graph optimization using the matrix representation in SE-Sync [**7**] except for the weight factors. In addition, (SE-MLE) can also be constructed as a specialized case of SE-Sync's [**7**] measurement model if we interpret pose-landmark measurements as pose-pose measurements whose rotational weight factors are zero.

In the next subsection, we will simplify (MLE) to quadratic programming on the product of unit complex numbers $\mathbb{C}_1^n$.

3.5.2. Problem Simplification

The simplification of (MLE) is similar to that of [**7**, Appendix B], the difference of which is that ours uses the complex number representation while [**7**] uses the matrix representation and ours has landmarks involved while [**7**] does not.

For notational convenience, we define $z_{ji}=\overline{z}_{ij}$, $\kappa_{ji}=\kappa_{ij}$ and $\tau_{ji}=\tau_{ij}$, and (MLE) can be reformulated as

$$\text{(P)} \qquad \min_{\xi\in\mathbb{C}^{n'}\times\mathbb{C}^n\times\mathbb{C}_1^n} \xi^H\widetilde{\Gamma}\xi$$

in which

$$\xi \triangleq \begin{bmatrix} s_1 & \cdots & s_{n'} & c_1 & \cdots & c_n & z_1 & \cdots & z_n \end{bmatrix}^\top.$$

In (P), $\widetilde{\Gamma}$ is a $(2n+n')$-by-$(2n+n')$ Hermitian matrix

$$\widetilde{\Gamma} \triangleq \begin{bmatrix} \Sigma^s & U & \widetilde{N} \\ * & L(W^c)+\Sigma^c & \widetilde{E} \\ * & * & L(\widetilde{G}^z)+\widetilde{\Sigma}^z \end{bmatrix}, \tag{3.11}$$

in which $\widetilde{\Sigma}^s \in \mathbb{H}^{n'}$, $\widetilde{U} \in \mathbb{C}^{n'\times n}$, $\widetilde{N} \in \mathbb{C}^{n'\times n}$, $L(W^c) \in \mathbb{H}^n$, $\widetilde{\Sigma}^c \in \mathbb{H}^n$, $\widetilde{E} \in \mathbb{C}^{n\times n}$, $L(\widetilde{G}^z) \in \mathbb{H}^n$ and $\widetilde{\Sigma}^z \in \mathbb{H}^n$ are defined as

$$[\Sigma^s]_{ij} \triangleq \begin{cases} \sum\limits_{(k,i)\in\overrightarrow{\mathcal{E}'}} \nu_{ki}, & i=j, \\ 0 & \text{otherwise}, \end{cases}$$

$$[U]_{ij} \triangleq \begin{cases} -\nu_{ji}, & (j,i)\in\overrightarrow{\mathcal{E}'}, \\ 0 & \text{otherwise}, \end{cases}$$

$$[\widetilde{N}^s]_{ij} \triangleq \begin{cases} -\nu_{ji}\tilde{s}_{ji}, & (j,i)\in\overrightarrow{\mathcal{E}'}, \\ 0 & \text{otherwise}, \end{cases}$$

$$[L(W^c)]_{ij} \triangleq \begin{cases} \sum\limits_{(i,k)\in\mathcal{E}} \tau_{ik}, & i=j, \\ -\tau_{ij}, & (i,j)\in\mathcal{E}, \\ 0 & \text{otherwise}, \end{cases}$$

$$[\Sigma^c]_{ij} \triangleq \begin{cases} \sum\limits_{(i,k)\in\overrightarrow{\mathcal{E}'}} \nu_{ik}, & i=j, \\ 0 & \text{otherwise}, \end{cases}$$

$$[\widetilde{E}]_{ij} \triangleq \begin{cases} \sum\limits_{(i,k)\in\overrightarrow{\mathcal{E}}} \tau_{ik}\tilde{c}_{ik} + \sum\limits_{(i,k)\in\overrightarrow{\mathcal{E}'}} \nu_{ik}\tilde{s}_{ik}, & i=j, \\ -\tau_{ij}\tilde{c}_{ji}, & (j,i)\in\overrightarrow{\mathcal{E}}, \\ 0 & \text{otherwise}, \end{cases}$$

$$[L(\widetilde{G}^z)]_{ij} \triangleq \begin{cases} \sum\limits_{(i,k)\in\mathcal{E}} \kappa_{ik}, & i=j, \\ -\kappa_{ij}\tilde{z}_{ji}, & (i,j)\in\mathcal{E}, \\ 0 & \text{otherwise}, \end{cases}$$

$$[\widetilde{\Sigma}^z)]_{ij} \triangleq \begin{cases} \sum\limits_{(i,k)\in\overrightarrow{\mathcal{E}}} \tau_{ik}|\tilde{c}_{ik}|^2 + \sum\limits_{(i,k)\in\overrightarrow{\mathcal{E}'}} \nu_{ik}|\tilde{s}_{ik}|^2, & i=j \\ 0 & \text{otherwise}, \end{cases}$$

respectively.

It is possible to marginalize the translational states and landmarks and reformulate planar graph-based SLAM as an optimization problem on the rotational states only, which has been used in **[7, 69, 93]** to improve the computational efficiency. In a similar way, if rotational states $z \triangleq \begin{bmatrix} z_1 & \cdots & z_n \end{bmatrix}^\top \in \mathbb{C}_1^n$ are known, (P) is reduced to unconstrained complex quadratic programming on translational states $c \triangleq \begin{bmatrix} c_1 & \cdots & c_n \end{bmatrix}^\top \in \mathbb{C}^n$ and

landmark positions $s \triangleq \begin{bmatrix} s_1 & \cdots & s_{n'} \end{bmatrix}^\top \in \mathbb{C}^w$:

$$\min_{\beta \in \mathbb{C}^{n'+n}} \ \beta^H \Lambda \beta + 2\big\langle \beta, \widetilde{\Theta} z \big\rangle + \underbrace{z^H L(\widetilde{G}^z) z + z^H \widetilde{\Sigma}^z z}_{\text{constant}}, \tag{3.12}$$

in which $\beta \triangleq \begin{bmatrix} s^\top & c^\top \end{bmatrix}^\top \in \mathbb{C}^{n+n'}$, $\widetilde{\Theta} \triangleq \begin{bmatrix} \widetilde{N} \\ \widetilde{E} \end{bmatrix} \in \mathbb{C}^{(n+n') \times n}$, and $\Lambda \triangleq \begin{bmatrix} \Sigma^s & U \\ * & L(W^c) + \Sigma^c \end{bmatrix} \in \mathbb{H}^{n+n'}$. It can be shown that according to [**94**, Proposition 4.2][2], one of the optimal solutions to Eq. (3.12) is

$$\beta = -\Lambda^\dagger \widetilde{\Theta} z. \tag{3.13}$$

Substituting Eq. (3.13) into (P) and simplifying the resulting equation, we obtain the complex quadratic programming on the product of unit complex numbers $\mathbb{C}_1^n$ as follows

$$\min_{z \in \mathbb{C}_1^n} z^H M z, \tag{3.14}$$

in which $M = L(\widetilde{G}^z) + \widetilde{\Sigma}^z - \widetilde{\Theta}^H \Lambda^\dagger \widetilde{\Theta} \succeq 0$.

Furthermore, let $\Omega \in \mathbb{R}^{(m+m') \times (m+m')}$ be the diagonal matrix indexed by $e \in \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}}'$ and $e' \in \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}}'$ whose (e, e')-element is given by

$$[\Omega]_{ee'} \triangleq \begin{cases} \nu_e, & e = e' \text{ and } e \in \overrightarrow{\mathcal{E}'}, \\ \tau_e, & e = e' \text{ and } e \in \overrightarrow{\mathcal{E}}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.15}$$

[2]It should be noted that [**94**, Proposition 4.2] was originally derived for real matrices, however, the results can be generalized to complex matrices as well.

in which ν_e and $\tau_e \in \mathbb{R}$ are the precisions of the landmark positional observations and the translational observations as given in Eqs. (3.10c) and (3.10a), respectively; and let $\widetilde{T} \in \mathbb{C}^{(m+m')\times n}$ be the matrix indexed by $e \in \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}'}$ and $k \in \mathcal{V} \cup \mathcal{V}'$ whose $(e,\, k)$-element is given by

$$[\widetilde{T}]_{ek} \triangleq \begin{cases} -\tilde{s}_{kj}, & e = (k,\, j) \in \overrightarrow{\mathcal{E}'}, \\ -\tilde{c}_{kj}, & e = (k,\, j) \in \overrightarrow{\mathcal{E}}, \\ 0, & \text{otherwise;} \end{cases} \tag{3.16}$$

and let $A(\overrightarrow{\mathcal{G}}) \in \mathbb{R}^{n\times m}$ to the matrix indexed by $k \in \mathcal{V} \cup \mathcal{V}'$ and $e \in \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}'}$ whose $(k,\, e)$-element is given by

$$[A(\overrightarrow{\mathcal{G}})]_{ke} = \begin{cases} 1, & e = (i,\, k) \in \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}'}, \\ -1, & e = (k,\, j) \in \overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}'}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.17}$$

In addition, without loss of any generality, we also introduce the ordering over $\overrightarrow{\mathcal{E}} \cup \overrightarrow{\mathcal{E}'}$ and $\mathcal{V} \cup \mathcal{V}'$ such that $e' \in \overrightarrow{\mathcal{E}'}$ precedes $e \in \overrightarrow{\mathcal{E}}$ and $k' \in \mathcal{V}'$ precedes $k \in \mathcal{V}$. As a result of Eqs. (3.15) to (3.17), $M = L(\widetilde{G}^z) + \widetilde{\Sigma}^z - \widetilde{\Theta}^H \Lambda^\dagger \widetilde{\Theta}$ can be rewritten as

$$M = L(\widetilde{G}^z) + \widetilde{T}^H \Omega^{\frac{1}{2}} \Pi \Omega^{\frac{1}{2}} \widetilde{T}, \tag{3.18}$$

in which $\Pi \in \mathbb{R}^{(m+m')\times(m+m')}$ is the matrix of the orthogonal projection operator $\pi : \mathbb{C}^{m+m'} \to \ker(A(\overrightarrow{\mathcal{G}})\Omega^{\frac{1}{2}})$ onto the kernel of $A(\overrightarrow{\mathcal{G}})\Omega^{\frac{1}{2}}$. Therefore, Eq. (3.14) is equivalent to

$$\text{(QP)} \qquad \begin{aligned} &\min_{z\in\mathbb{C}_1^n} \operatorname{trace}(Mzz^H), \\ &M = L(\widetilde{G}^z) + \widetilde{T}^H \Omega^{\frac{1}{2}} \Pi \Omega^{\frac{1}{2}} \widetilde{T}. \end{aligned}$$

The detailed derivation of (QP) is presented in Section 3.5.3.

3.5.3. The Derivation of (QP)

In this subsection, we derive (QP) following a similar procedure of [**7**, Appendix B] even though ours uses the complex number representation and has landmarks involved.

It is straightforward to rewrite (MLE) as

$$\min_{\substack{s\in\mathbb{C}^n,\\ c\in\mathbb{C}^n,\, z\in\mathbb{C}_1^n}} \left\| B \begin{bmatrix} s \\ c \\ z \end{bmatrix} \right\|_2^2 \tag{3.19}$$

in which

$$B \triangleq \begin{bmatrix} B_1 & B_2 \\ \mathbf{0} & B_3 \end{bmatrix} \in \mathbb{C}^{(2m+m')\times(2n+n')}.$$

Here $B_1 \in \mathbb{R}^{(m+m')\times(n+n')}$, $B_2 \in \mathbb{R}^{(m+m')\times n}$ and $B_3 \in \mathbb{C}^{m\times n}$ are given as

$$[B_1]_{ek} = \begin{cases} \sqrt{\nu_{ik}}, & e = (i,\, k) \in \overrightarrow{\mathcal{E}'}, \\ -\sqrt{\nu_{kj}}, & e = (k,\, j) \in \overrightarrow{\mathcal{E}'}, \\ \sqrt{\tau_{ik}}, & e = (i,\, k) \in \overrightarrow{\mathcal{E}}, \\ -\sqrt{\tau_{kj}}, & e = (k,\, j) \in \overrightarrow{\mathcal{E}}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.20a}$$

(3.20b)
$$[B_2]_{ek} = \begin{cases} -\sqrt{\nu_{kj}}\tilde{s}_{kj}, & e = (k,\, j) \in \overrightarrow{\mathcal{E}'}, \\ -\sqrt{\tau_{kj}}\tilde{c}_{kj}, & e = (k,\, j) \in \overrightarrow{\mathcal{E}}, \\ 0, & \text{otherwise}, \end{cases}$$

and

(3.20c)
$$[B_3]_{ek} = \begin{cases} -\sqrt{\kappa_{kj}}\tilde{z}_{kj}, & e = (k,\, j) \in \overrightarrow{\mathcal{E}}, \\ \sqrt{\kappa_{ik}}, & e = (i,\, k) \in \overrightarrow{\mathcal{E}}, \\ 0, & \text{otherwise}, \end{cases}$$

respectively. Since (P) is also equivalent to (3.12), it can be concluded that

(3.21a)
$$B_1^H B_1 = \Lambda,$$

(3.21b)
$$B_1^H B_2 = \widetilde{\Theta},$$

(3.21c)
$$B_2^H B_2 = \widetilde{\Sigma}^z,$$

(3.21d)
$$B_3^H B_3 = L(\widetilde{G}^z)$$

in which Λ, $\widetilde{\Theta}$, $\widetilde{\Sigma}$ and $L(\widetilde{G}^z)$ are as defined in (3.12). If we let $M^\sigma \triangleq \widetilde{\Sigma}^z - \widetilde{\Theta}^H \Lambda^\dagger \widetilde{\Theta}$, then from Eqs. (3.21a) to (3.21d), we obtain

(3.22)
$$\begin{aligned} M^\sigma &= B_2^H B_2 - B_2^H B_1 (B_1^H B_1)^\dagger B_1^H B_2 \\ &= B_2^H \left(\mathbf{I} - B_1 (B_1^H B_1)^\dagger B_1^H \right) B_2, \end{aligned}$$

in which B_1, B_2 and B_3 are defined as Eqs. (3.20a) to (3.20c). It should be noted that we might rewrite B_1 and B_2 as

$$B_1 = \Omega^{\frac{1}{2}} A^\top, \qquad B_2 = \Omega^{\frac{1}{2}} \widetilde{T}, \tag{3.23}$$

in which $A \triangleq A(\overrightarrow{\mathcal{G}})$ and $\widetilde{T}$ are given by Eqs. (3.17) and (3.16), respectively. Substituting Eq. (3.23) into Eq. (3.22), we obtain

$$\begin{aligned} M^\sigma &= B_2^H \left(\mathbf{I} - B_1 (B_1^H B_1)^\dagger B_1^H\right) B_2 \\ &= \widetilde{T}^H \Omega^{\frac{1}{2}} \Pi \Omega^{\frac{1}{2}} \widetilde{T}, \end{aligned} \tag{3.24}$$

in which

$$\Pi = \mathbf{I} - \Omega^{\frac{1}{2}} A^\top \left(A \Omega A^\top\right)^\dagger A \Omega^{\frac{1}{2}} \in \mathbb{R}^{(m+m')\times(m+m')}.$$

As a result, it can be concluded that

$$M = L(\widetilde{G}^z) + M^\sigma = L(\widetilde{G}^z) + \widetilde{T}^H \Omega^{\frac{1}{2}} \Pi \Omega^{\frac{1}{2}} \widetilde{T}.$$

Furthermore, it is known that $X^\top (X X^\top)^\dagger = X^\dagger$ for any matrix X, then we further obtain

$$\begin{aligned} \Pi &= \mathbf{I} - \Omega^{\frac{1}{2}} A^\top \left(A \Omega A^\top\right)^\dagger A \Omega^{\frac{1}{2}} \\ &= \mathbf{I} - \left(A \Omega^{\frac{1}{2}}\right)^\dagger A \Omega^{\frac{1}{2}}, \end{aligned} \tag{3.25}$$

which according to [**95**, Chapter 5.13] is the matrix of orthogonal projection operator $\pi : \mathbb{C}^{m+m'} \to \ker(A(\overrightarrow{\mathcal{G}})\Omega^{\frac{1}{2}})$ onto the kernel space of $A\Omega^{\frac{1}{2}}$. In addition, similar to [**7**, Appendix B.2], it is possible to further decompose Π in terms of sparse matrices and their inverse for efficient computation even though Π is in general a dense matrix.

In the next section, we will relax (QP) to complex semidefinite programming and show that the semidefinite relaxation is tight as long as the noise magnitude is below a certain threshold.

3.6. The Semidefinite Relaxation

In a similar way to **[72, 79, 80]**, it is straightforward to relax (QP) to

$$\text{(SDP)} \qquad \begin{aligned} &\min_{X\in\mathbb{H}^n} \left\langle M, X \right\rangle \\ &\text{s.t.} \quad X \succeq 0, \quad \operatorname{diag}(X) = \mathbf{1}. \end{aligned}$$

It should be noted that if $\hat{X} \in \mathbb{H}^n$ has rank one and solves (SDP), then a solution $\hat{z} \in \mathbb{C}_1^n$ to (QP) can be exactly recovered from $\hat{X}$ through singular value decomposition with which we have $\hat{X} = \hat{z}\hat{z}^H$.

In this chapter, it is without loss of any generality to assume that all the manifolds are Riemannian submanifolds of Euclidean space **[90]**, whose differential geometric properties, e.g., Riemannian gradients and Riemannian Hessians, are defined accordingly.

In the rest of section, we will analyze and derive the conditions for the optimality of (QP) and (SDP), and conditions for the tight relaxation of (SDP), all the proofs of which can be found in Section 3.10.

From **[90]**, the necessary conditions for the local optimality of (QP) can be well characterized in terms of the Riemannian gradients and Hessians.

Lemma 3.1. If $\hat{z} \in \mathbb{C}_1^n$ is a local optimum of (QP), then there exists a real diagonal matrix $\hat{\Lambda} \triangleq \Re\{\operatorname{ddiag}(M\hat{z}\hat{z}^H)\} \in \mathbb{R}^{n\times n}$ such that $\hat{S} \triangleq M - \hat{\Lambda} \in \mathbb{H}^n$ satisfies the following conditions:

(1) $\hat{S}\hat{z} = \mathbf{0}$;

(2) $\langle \dot{z}, \hat{S}\dot{z} \rangle \geq 0$ for all $\dot{z} \in T_{\hat{z}}\mathbb{C}_1^n$.

If $\hat{z}$ satisfies (1), it is a first-order critical point, and if $\hat{z}$ satisfies (1) and (2), it is a second-order critical point.

Proof. See Section 3.10.1. □

Since (SDP) is convex and the identity matrix $\mathbf{I} \in \mathbb{C}^{n\times n}$ is strictly feasible, the sufficient and necessary conditions for the global optimality of (SDP) can be derived in terms of the Karush-Kuhn-Tucker (KKT) conditions.

Lemma 3.2. A Hermitian matrix $\hat{X} \in \mathbb{H}^n$ is a global optimum of (SDP) if and only if there exists $\hat{S} \in \mathbb{H}^n$ such that the following conditions hold:

(1) $\operatorname{diag}(\hat{X}) = \mathbf{1}$;

(2) $\hat{X} \succeq 0$;

(3) $\hat{S}\hat{X} = \mathbf{0}$;

(4) $M - \hat{S}$ is real diagonal;

(5) $\hat{S} \succeq 0$.

Furthermore, if $\operatorname{rank}(\hat{S}) = n - 1$, then $\hat{X}$ has rank one and is the unique global optimum of (SDP).

Proof. See Section 3.10.2. □

As a result of Lemmas 3.1 and 3.2, we obtain the sufficient conditions for the exact recovery of (QP) from (SDP).

Lemma 3.3. If $\hat{z} \in \mathbb{C}_1^n$ is a first-order critical point of (QP) and $\hat{S} = M - \hat{\Lambda} \succeq 0$ in which $\hat{\Lambda} = \Re\{\text{ddiag}(M\hat{z}\hat{z}^H)\}$, then $\hat{z}$ is a global optimum of (QP) and $\hat{X} = \hat{z}\hat{z}^H$ is a global optimum of (SDP). Moreover, if $\text{rank}(\hat{S}) = n - 1$, then $\hat{X}$ is the unique optimum of (SDP).

Proof. See Section 3.10.3. □

Lemma 3.3 gives sufficient conditions to check whether (SDP) is a tight relaxation of (QP). As a matter of fact, if the measurement noise is not too large, it is guaranteed that (SDP) is always a tight relaxation of (QP) as the following proposition states.

Proposition 3.6.1. Let $\underline{M} \in \mathbb{H}^n$ be the data matrix of the form Eq. (3.18) that is constructed with the true (latent) pose-pose measurements $\underline{\rho}_{ij} = (\underline{c}_{ij}, \underline{z}_{ij})$ and pose-landmark measurements $\underline{s}_{ij}$, then there exists a constant $\gamma = \gamma(\underline{M}) > 0$ such that if $\|M - \underline{M}\|_2 < \gamma$, then (SDP) attains the unique global optimum at $\hat{X} = \hat{z}\hat{z}^H \in \mathbb{H}^n$, in which $\hat{z} \in \mathbb{C}_1^n$ is a global optimum of (QP).

Proof. See Section 3.10.4. □

Lemma 3.3 verifies the tightness of the complex semidefinite relaxation and Proposition 3.6.1 guarantees that the tightness of the complex semidefinite relaxation, which makes (SDP) certifiably correct for graph-based SLAM. It should be noted that similar results to Lemma 3.3 and Proposition 3.6.1 have been presented for synchronization problems on general special Euclidean groups using the matrix representation in [**7**] and for phase synchronization using the complex number representation in [**79**].

In spite of the tightness of the semidefinite relaxation of planar graph-based SLAM, solving large-scale complex semidefinite programming remains challenging and time-consuming.

In the next section, we will further relax (SDP) as a series of rank-restricted complex semidefinite programming such that (SDP) can be efficiently solved with the Riemannian staircase optimization.

3.7. The CPL-SLAM Algorithm

In this section, we show that it is possible to recast (SDP) as Riemannian optimization on complex oblique manifolds, which is one of the most important contributions of this chapter. A brief introduction to the complex oblique manifold has been given in Section 3.4, and it is also helpful to read [**91**] that is about the real oblique manifold.

3.7.1. Riemannian Staircase Optimization

In general, interior point methods to solve (SDP) take polynomial time, which, however, may still be slow when the polynomial exponent is large. Instead of solving (SDP) directly, Boumal et al. found that (SDP) can be relaxed to a series of rank-restricted complex semidefinite programing [**84**]:

$$(r\text{-SDP}) \qquad \min_{Y \in \mathrm{OB}(r,n)} \operatorname{trace}(MYY^H)$$

in which

$$\mathrm{OB}(r,n) \triangleq \{Y \in \mathbb{C}^{n\times r} | \operatorname{ddiag}(YY^H) = \mathbf{I}\}$$

is the complex oblique manifold. It should be noted that (r-SDP) can be a tight relaxation of (SDP) if some conditions are met as stated in Propositions 3.7.1 and 3.7.2, whose proofs are immediate from [**84**, Theorem 2].

Proposition 3.7.1. If $\hat{Y} \in \mathrm{OB}(r,n)$ is rank-deficient and second-order critical for (r-SDP), then it is globally optimal for (r-SDP) and $\hat{X} = \hat{Y}\hat{Y}^H \in \mathbb{H}^n$ is globally optimal for (SDP).

Proposition 3.7.2. If $r \geq \lceil \sqrt{n} \rceil$, then for almost all $M \in \mathbb{C}^{n\times n}$, every first-order critical $\hat{Y} \in \mathrm{OB}(r,n)$ for (r-SDP) is rank-deficient.

Propositions 3.7.1 and 3.7.2 are referred as the Burer-Monteiro guarantees for smooth semidefinite programming [**84**] that apply to a number of classic estimation problems. From Propositions 3.7.1 and 3.7.2, it can be concluded that (SDP) is equivalent to successively solving (r-SDP) with the Riemannian trust region (RTR) method [**96**] for $2 \leq r_1 < r_2 < \cdots < r_k \leq n+1$ until a rank-deficient second-order critical point is found, and such a method to solve semidefinite programming is referred as the Riemannian staircase optimization (Algorithm 6) [**84**, **97**]. In [**7**, **79**, **97**], the Riemannian staircase optimization has been used to solve a number of semidefinite relaxations of synchronization problems. In addition, it is known that the RTR method solves (r-SDP) locally in polynomial time [**84**, Proposition 3]. In contrast to using interior point methods to solve (SDP) directly, the Riemannian staircase optimization using the RTR method is empirically orders of magnitude faster in solving large-scale smooth semidefinite programming.

As shown in Algorithm 7, the solution rounding of an optimum of $Y^* \in \mathrm{OB}(r,n)$ of (r-SDP) is simply to assign $\hat{z} = \begin{bmatrix}\hat{z}_1 & \cdots & \hat{z}_n\end{bmatrix} \in \mathbb{C}^n$ to be the left-singular vector of Y^* that is associated with the greatest singular value, and then normalize each z_i to get $\hat{z} \in \mathbb{C}_1^n$. The solution rounding algorithm is exact if $\mathrm{rank}(Y^*) = 1$. Moreover, it should be noted that the solution rounding algorithm can recover the global optimum $\hat{z}^* \in \mathbb{C}_1^n$ from Y^* as long as the exactness of (SDP) holds and $X = Y^*Y^{*H}$ solves (SDP).

Algorithm 6 The Riemannian staircase optimization (RSO)

Input: Integers $2 \leq r_0 < r_1 < \cdots < r_k \leq n+1$; an initial iterate $z_0 \in \mathbb{C}_1^n$
$Y_0 = \begin{bmatrix} \hat{z}_0 & \mathbf{0} \end{bmatrix} \in \mathrm{OB}(r_0, n)$
for $i = 1 \to k$ **do**
    Implement the Riemannian optimization to solve

$$Y_i^* = \arg \min_{Y \in \mathrm{OB}(r_i, n)} \mathrm{trace}(MYY^H)$$

    locally with Y_i as an initial guess
    if $\mathrm{rank}(Y_i^*) < r_i$ **then**
        return $Y_i^* \in \mathrm{OB}(r_i, n)$
    else
        $Y_{i+1} = \begin{bmatrix} \hat{Y}_i & \mathbf{0} \end{bmatrix} \in \mathrm{OB}(r_{i+1}, n)$
    end if
end for
return $Y_i^* \in \mathrm{OB}(r_k, n)$

Algorithm 7 The rounding procedure for solutions of (r-SDP)

Input: An optimum $Y^* \in \mathrm{OB}(r, n)$ to (r-SDP)
Assign $\hat{z} = \begin{bmatrix} \hat{z}_1 & \cdots & \hat{z}_n \end{bmatrix}^\top \in \mathbb{C}^n$ to be the left-singular vector of Y^* that is associated with the greatest singular value
for $i = 1 \to n$ **do**
    $\hat{z}_i = \dfrac{\hat{z}_i}{|\hat{z}_i|}$
end for
return $\hat{z} \in \mathbb{C}_1^n$

From algorithms of Riemannian staircase optimization (Algorithm 6) and solution rounding (Algorithm 7), the proposed CPL-SLAM algorithm for planar graph-based SLAM is as shown in Algorithm 8, which follows a similar procedure to SE-Sync [**7**]. It should be noted that Lemmas 3.1 to 3.3 can be used to certify the global optimality of the solution, and Propositions 3.6.1, 3.7.1 and 3.7.2 indicate that the CPL-SLAM algorithm is expected to retrieve the globally optimal solution to the planar graph-based SLAM as long as the noise magnitude is below a certain threshold. Therefore, it can be concluded that CPL-SLAM is certifiably correct.

Algorithm 8 The CPL-SLAM algorithm

1: **Input**: Integers $2 \leq r_0 < r_1 < \cdots < r_k \leq n+1$; an initial iterate $z_0 \in \mathbb{C}_1^n$
2: Implement Algorithm 6 to compute an optimum $Y^* \in \mathrm{OB}(r, n)$
3: Implement Algorithm 7 to compute rotational states $\hat{z} \in \mathbb{C}_1^n$
4: Implement Eq. (3.13) to compute translational states $\hat{c} \in \mathbb{C}^n$
5: **return** $\hat{z} \in \mathbb{C}_1^n$ and $\hat{c} \in \mathbb{C}^n$

3.7.2. The Preconditioner for the CPL-SLAM Algorithm

As SE-Sync [**7**] and Cartan-Sync [**75**], CPL-SLAM uses the trust-region method on Riemannian manifolds that relies on the truncated conjugated gradient (TCG) method to evaluate the descent direction [**96**]. The TCG method iteratively solves linear equations and improves the solution to necessary accuracy within finite iterations, which is usually faster than direct methods. In general, the TCG method needs a preconditioner to accelerate the convergence. Even though the choice of preconditioner for graph-based SLAM without landmarks is immediate [**7**], there is still a lack of a suitable preconditioner for graph-based SLAM with landmarks. To address this issue, we also propose a preconditioner for graph-based SLAM with landmarks as follows.

Similar to [**7**,**75**,**98**], instead of factorizing the Riemannian Hessian matrix $\mathrm{Hess}\, F(z) \in \mathbb{C}^{n\times n}$ in Eq. (3.33) to explicitly evaluate the descent direction, the Riemannian trust region (RTR) method in CPL-SLAM leverages the truncated conjugated gradient (TCG) method to approximate the descent direction with necessary accuracy. Though the TCG method is guaranteed to converge to the true solution within finite iterations, the rates of the convergence is closely related with the preconditioner $\mathrm{Precon}(\mathrm{Hess}\, F(z)) \in \mathbb{C}^{n\times n}$ that is used to approximate $\mathrm{Hess}\, F(z)$ and iteratively solve

$$\mathrm{Precon}(\mathrm{Hess}\, F(z)) \cdot a = b$$

to evaluate the descent direction, in which a, $b \in \mathbb{C}^n$.

In graph-based SLAM, several preconditioners have been proposed for the TCG method [**75**,**98**]. For CPL-SLAM, an immediate choice of the preconditioner Precon(Hess $F(z)$) is $L(\widetilde{G}^z) + \widetilde{\Sigma}^z$, which is the submatrix of $\widetilde{\Gamma}$ in Eq. (3.11) that corresponds to the rotational states,[3] and such a preconditioner works well for planar graph-based SLAM without landmarks. However, the preconditioner of $L(\widetilde{G}^z) + \widetilde{\Sigma}^z$ suffers slow convergence for planar graph-based SLAM with landmarks since the submatrix $L(\widetilde{G}^z) + \widetilde{\Sigma}^z$ loses information of pose-landmark measurements and results in a bad approximation of Hess $F(z)$.

In contrast to $L(\widetilde{G}^z) + \widetilde{\Sigma}^z$ that only captures the information of pose-pose measurements, the matrix

$$M = L(\widetilde{G}^z) + \widetilde{T}^H \Omega^{\frac{1}{2}} \Pi \Omega^{\frac{1}{2}} \widetilde{T} \in \mathbb{C}^{n \times n} \tag{3.26}$$

in (QP) implicitly but properly keeps the information of both pose-pose measurements and pose-landmark measurements. However, there is no exact expression of M and we need to evaluate the equation above to factorize M. Furthermore, since M can be a dense matrix, the resulting factorization of M might be inefficient to solve

$$M \cdot a = b, \tag{3.27}$$

which affects the performance of the TCG method. As a result, we need some other methods rather than evaluate and factorize M explicitly to solve Eq. (3.27).

[3] A similar preconditioner is also used in SE-Sync [**98**].

It should be noted that the solution to

$$\min_{x\in\mathbb{C}^n} \frac{1}{2}\big\langle x, Mx\big\rangle - \big\langle x, b\big\rangle, \tag{3.28}$$

is also a solution to Eq. (3.27). From Eqs. (3.11) to (3.14), it is straightforward to show that Eq. (3.28) is equivalent to

$$\min_{x\in\mathbb{C}^n,\, x'\in\mathbb{C}^{n+n'}} \frac{1}{2}\Big\langle \begin{bmatrix} x' \\ x \end{bmatrix}, \widetilde{\Gamma} \begin{bmatrix} x' \\ x \end{bmatrix} \Big\rangle - \Big\langle \begin{bmatrix} x' \\ x \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ b \end{bmatrix} \Big\rangle, \tag{3.29}$$

and the solution $\begin{bmatrix} a' \\ a \end{bmatrix} \in \mathbb{C}^{2n+n'}$ to Eq. (3.29) can be computed in closed form as

$$\widetilde{\Gamma} \begin{bmatrix} a' \\ a \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ b \end{bmatrix}, \tag{3.30}$$

or equivalently,

$$\begin{bmatrix} a' \\ a \end{bmatrix} = \widetilde{\Gamma}^{\dagger} \begin{bmatrix} \mathbf{0} \\ b \end{bmatrix}. \tag{3.31}$$

It is by definition that $a \in \mathbb{C}^n$ in Eqs. (3.30) and (3.31) is also a solution to Eqs. (3.27) and (3.28). Therefore, we might factorize $\widetilde{\Gamma}$ and solve Eqs. (3.30) and (3.31) instead so as to solve Eq. (3.27). In Eqs. (3.14), (3.27) and (3.28), we construct the dense data matrix M using Eq. (3.13) to reduce the dimension of the optimization problem with no information loss. In Eqs. (3.30) and (3.31), on the other hand, we essentially reverse the operation of Eq. (3.13) to recover the sparse matrix $\widetilde{\Gamma}$ by augmenting the dimension.

Since $\widetilde{\Gamma}$ is a sparse matrix whose exact expression requires no extra computation, it is more efficient to exploit the sparsity of $\widetilde{\Gamma}$ to solve Eqs. (3.30) and (3.31) than factorize the dense data matrix M to solve Eqs. (3.27) and (3.28).

3.7.3. Discussion

Even though the positive semidefinite matrix is not explicitly formed in CPL-SLAM or SE-Sync to solve planar graph-based SLAM, it can be seen that CPL-SLAM using the complex number representation results in semidefinite relaxations of smaller size than [**7**, **72**]. In the semidefinite relaxation of CPL-SLAM, the $n \times n$ complex positive semidefinite matrix $X \in \mathbb{H}^n$ can be parameterized with n^2 real numbers, whereas the semidefinite relaxation in [**7**] using the matrix representation needs $2n^2 + n$ real numbers to parameterize the $2n \times 2n$ real positive semidefinite matrix, and that in [**72**] needs $4n^2$ real numbers to parameterize the $2n \times 2n$ complex positive semidefinite matrix.

It is obvious that the complex number representation is more concise than the matrix representation, and as a result, CPL-SLAM roughly requires half as much storage space as SE-Sync [**7**]. More importantly, as is discussed in Section 3.8, from both theoretical and empirical perspectives, the conciseness of the complex number representation reduces the computational cost a lot and renders the semidefinite relaxation much tighter, and thus, the resulting CPL-SLAM algorithm is much more efficient in numerical computation and much more robust to measurement noise than [**7**].

In contrast to the works of [**7**, **72**, **76**], CPL-SLAM is more general and more scalable. As mentioned before, CPL-SLAM is more efficient, tighter and more robust than SE-Sync [**7**]. Even though we use the same complex number representation as [**72**], our formulation

is simpler and only depends on rotational states $z \in \mathbb{C}_1^n$, whereas [**72**] involves both translational and rotational states $c \in \mathbb{C}^n$ and $z \in \mathbb{C}_1^n$. Moreover, [**72**] mainly focuses on the optimality verification of planar pose graph optimization, whereas ours not only works on optimality verification and obtains stronger theoretical results, but also presents more scalable algorithms to solve planar graph-based SLAM. In [**76**], the authors use bounded sum of squares programming to solve planar graph-based SLAM. Even though [**76**] always attains the globally optimal solution regardless of the measurement noise, it relies on sparse sum of squares programming, which, to our knowledge, has limited scalability for large-scale problems. As a result, CPL-SLAM can be expected to outperform [**76**] by several orders of magnitude in terms of computational time. Last but not least, except for [**76**], the works of [**7**, **72**] are designed for planar pose graph optimization or angular synchronization, whereas ours considers the planar graph-based SLAM that has both poses and landmarks.

3.8. Experiments

In this section, we implement CPL-SLAM on the simulated Tree datasets, simulated City datasets and a suite of large-scale 2D SLAM benchmark datasets with and without landmarks [**7**,**72**,**99**]. We compare CPL-SLAM with the popular state-of-the-art SE-Sync [**7**] and Powell's Dog-Leg method (PDL-GN) [**63**,**89**]. Even though the original algorithms of SE-Sync [**7**] are not designed for problems with landmarks, we extend SE-Sync following a similar procedure as CPL-SLAM. For the linear solvers to compute a descent direction, CPL-SLAM and SE-Sync [**7**] use the indirect and iterative truncated conjugate gradient

method, whereas PDL-GN [**63**, **89**] uses the sparse direct method. The C++ code of CPL-SLAM is available at `https://github.com/MurpheyLab/CPL-SLAM`.

All the experiments have been performed on a laptop with an Intel i7-8750H CPU and 32GB of RAM running Ubuntu 18.04 and using `g++ 7.8` as C++ compiler. We have done the computation on a single core of CPU. For all the experiments, we choose the initial rank to be $r_{\mathrm{SE}} = 3$ and $r_{\mathrm{CPL}} = 2$ for SE-Sync and CPL-SLAM, respectively, since we find that $r_{\mathrm{SE}} = 3$ and $r_{\mathrm{CPL}} = 2$ are in general good enough for SE-Sync and CPL-SLAM to solve planar graph-based SLAM given the noise levels in robotics and computer vision applications.

3.8.1. Tree Datasets

In this subsection, we evaluate the performance of CPL-SLAM, SE-Sync and PDL-GN on the simulated Tree datasets that are similar to tree10000 (Fig. 3.6k). A Tree dataset is consisted of 25×25 square grids in which each grid has side length of 1 m, and a robot trajectory of n poses along the rectilinear path of the square grid, and n' trees (landmarks) that are randomly distributed in the centre of some square grids. Odometric pose-pose measurements are available between each pair of sequential poses along the robot trajectory, whereas pose-landmark measurements between poses and trees that are close to each other are available with a probability of p_L; the pose-pose measurements $\tilde{\rho}_{ij} = (\tilde{c}_{ij}, \tilde{z}_{ij})$ and pose-landmark measurements $\tilde{s}_{ij}$ are generated from the noise models of Eq. (3.10). In our experiments, we investigate the performance of these algorithms by varying each parameter individually and the default values for these parameters are

chosen to be $n = 5000$, $n' = 250$, $p_L = 0.2$, $\tilde{c}_{ij}$ with an expected translational root-mean-squared error (RMSE) of $\sigma_t = 0.05$ m, $\tilde{z}_{ij}$ with an expected angular RMSE of $\sigma_R = 0.015\pi$ rad, and $\tilde{s}_{ij}$ with an expected positional RMSE of $\sigma_l = 0.05$ m.

For all the Tree datasets tested, CPL-SLAM, SE-Sync and PDL-GN all converge to the global optima when using the chordal initialization. As is shown in Fig. 3.1, it can be seen that CPL-SLAM is around $4 \sim 5$ times faster than SE-Sync and PDL-GN, whereas SE-Sync and PDL-GN are roughly as fast as each other.

The speed-up of CPL-SLAM over SE-Sync [**7**] in planar graph-based SLAM can be explained from several perspectives. 1) CPL-SLAM is more efficient for the objective and gradient evaluation, e.g., if the rank is $r_{\mathrm{SE}} = 3$ and $r_{\mathrm{CPL}} = 2$, CPL-SLAM only needs $\frac{1}{2} \sim \frac{2}{3}$ and $\frac{1}{4} \sim \frac{2}{3}$ operations of SE-Sync to evaluate the objective and gradient, respectively. 2) CPL-SLAM is more efficient in terms of the projection or retraction onto the manifold than SE-Sync – the projection map of CPL-SLAM is just to normalize n vectors, whereas that of SE-Sync has to compute n singular value decompositions, which is much more time consuming. 3) CPL-SLAM is more efficient for chordal initialization and solution rounding. 4) As a result of the conciseness of the complex number representation, the preconditioner used in CPL-SLAM has a better approximation the Hessian matrix than SE-Sync, and thus, has a faster convergence of the truncated conjugate gradient method that the Riemannian trust region method implements to evaluate the descent direction. Therefore, CPL-SLAM should be theoretically more efficient than SE-Sync, which is further confirmed by the results of the experiments.

Similar to [**66**, **69**, **100**], PDL-GN uses the Gauss-Newton method and might not perform well if there are large residues of the measurements and strong nonlinearities of

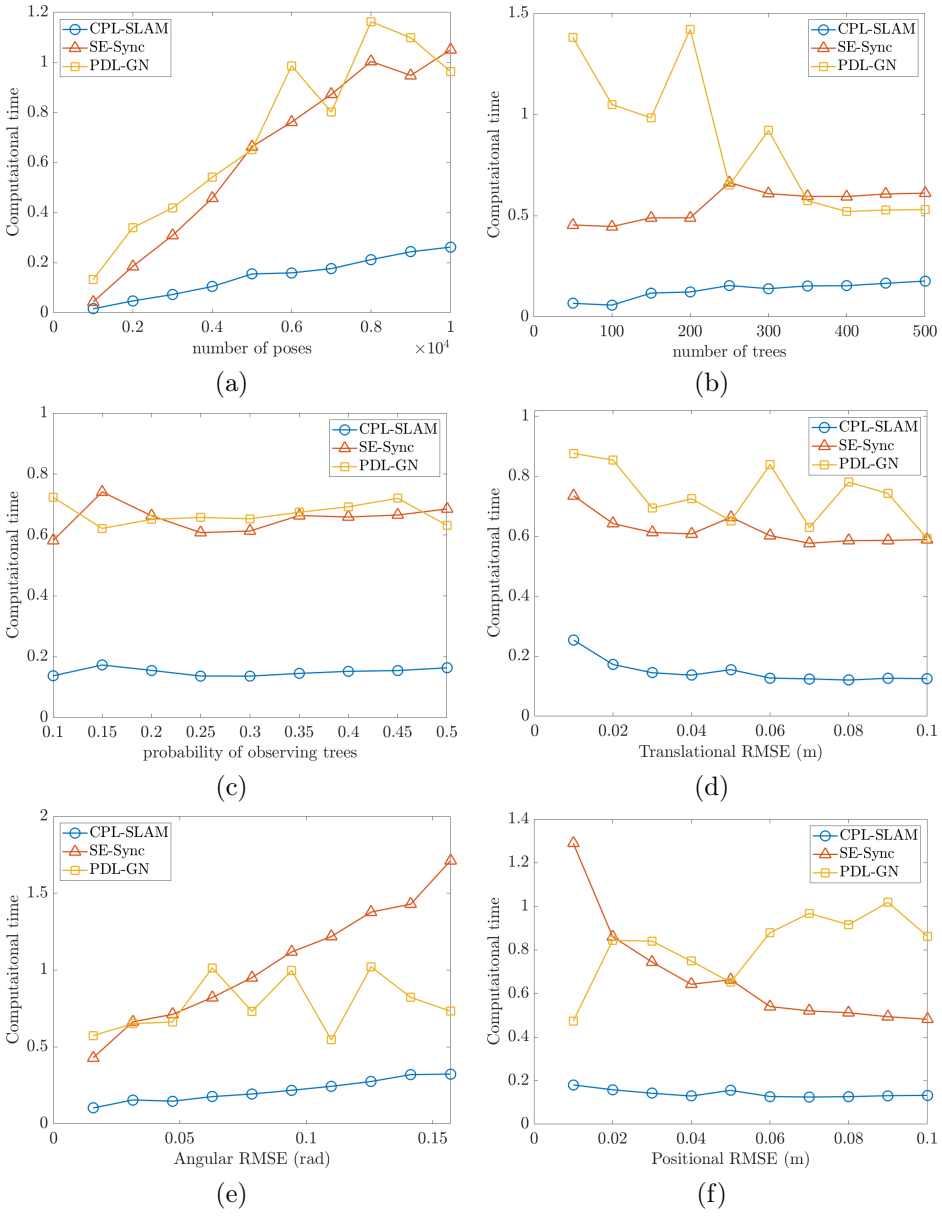


Figure 3.1. The computational time of CPL-SLAM, SE-Sync and PDL-GN on the Tree datasets with varying each parameter individually while keeping the other parameters to be default values. The chordal initialization is used for all the tests. The results of each varying parameter are the number of poses n in (a), the number of trees n' in (b), the probability of observing trees p_L in (c), translational RMSEs of σ_t in (d), angular RMSEs of σ_R in (e) and positional RMSEs of σ_l in (f). The default values are $n = 5000$, $n' = 250$, $p_L = 0.2$, $\sigma_t = 0.05$ m, $\sigma_R = 0.015\pi$ rad and $\sigma_l = 0.05$ m. For all the Tree datasets tested, it can be seen that CPL-SLAM is around $4 \sim 5$ times faster than SE-Sync and PDL-GN, whereas SE-Sync and PDL-GN are roughly as fast as each other.

the objective function [**63**,**89**], whereas CPL-SLAM uses the exact Hessian to compute the Newton direction, and thus, is expected to converge faster and have better efficiency. In addition, as mentioned before, when evaluating the descent direction, PDL-GN factorizes sparse matrices to solve linear equations. In contrast, CPL-SLAM makes use of the truncated conjugate gradient method as the linear solver, which might also improve the overall efficiency of CPL-SLAM. On the other hand, since the choice of linear solvers is critical for the efficiency of optimizers, there is a possibility to improve PDL-GN's efficiency if the truncated conjugate gradient method is used.

The performance of CPL-SLAM, SE-Sync and PDL-GN is also evaluated if they are not well initialized. When the odometric initialization is used, it can be seen from Fig. 3.2 that CPL-SLAM and SE-Sync converge to the global optima in spite of the poor initial guess, whereas PDL-GN gets stuck at the local optima and has much greater objective values.

As mentioned before, the convergence of CPL-SLAM and SE-Sync to global optima does not rely on initial guess since CPL-SLAM and SE-Sync essentially solve the semidefinite relaxation of graph-based SLAM and are guaranteed to attain the globally optimal solution as long as the magnitude of measurement noise is below a certain threshold. As a comparison, PDL-GN is a local search method whose performance is closely related with quality of initial guess, and thus the global optimum convergence of PDL-GN is usually not guaranteed even with low measurement noise.

3.8.2. City Datasets

In this subsection, we evaluate the tightness of CPL-SLAM on a series of simulated City datasets that are similar to city10000 (Fig. 3.6b) but with high measurement noise.

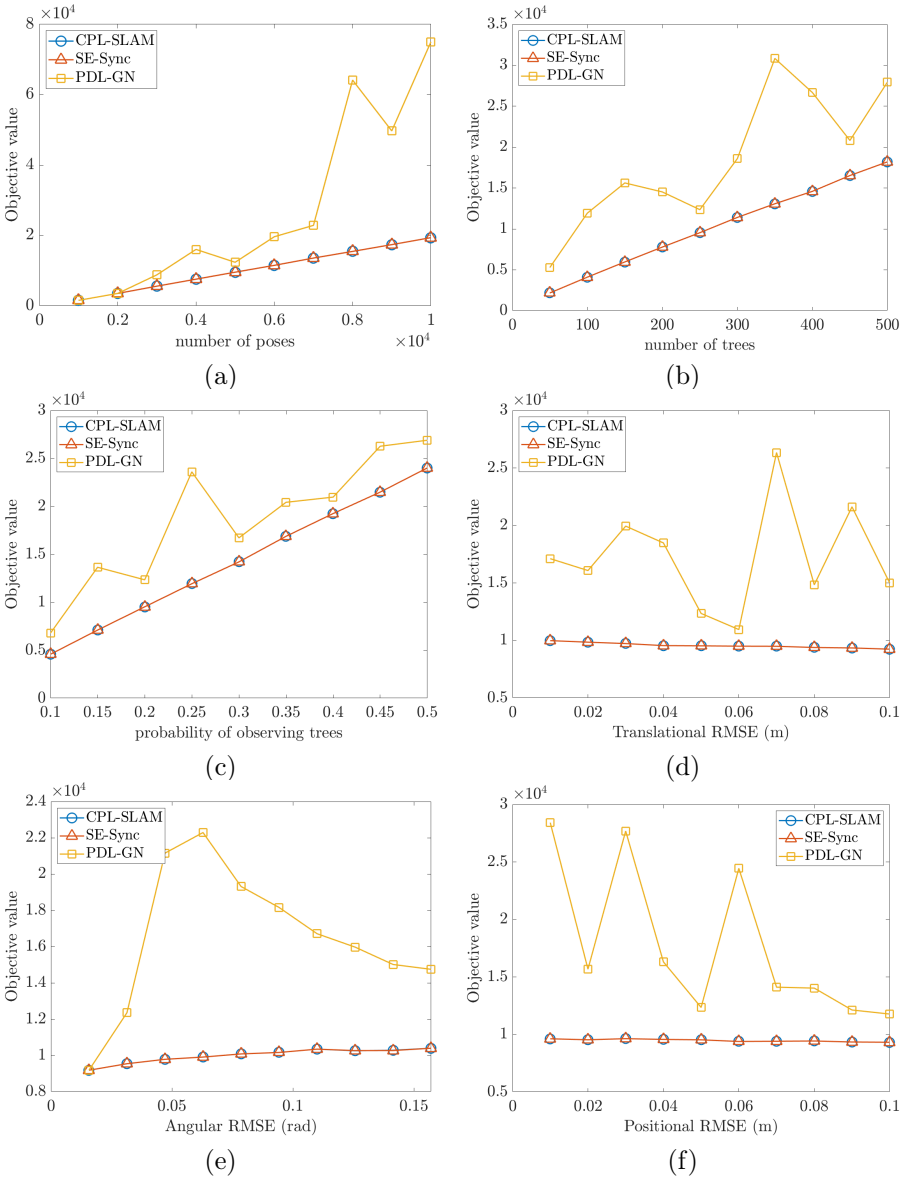


Figure 3.2. The objective of CPL-SLAM, SE-Sync and PDL-GN on the Tree datasets using the odometric initialization. In the experiments, we vary each parameter separately while the other parameters are set to be the default values. The results of each varying parameter are the number of poses n in (a), the number of trees n' in (b), the probability of observing trees p_L in (c), translational RMSEs of σ_t in (d), angular RMSEs of σ_R in (e) and positional RMSEs of σ_l in (f). The default values are $n = 5000$, $n' = 250$, $p_L = 0.2$, $\sigma_t = 0.05$ m, $\sigma_R = 0.015\pi$ rad and $\sigma_l = 0.05$ m. For all the Tree datasets tested, CPL-SLAM and SE-Sync converge to global optima despite poor initialization, whereas PDL-GN gets stuck at local optima.

As a basis for comparison, we also evaluate the tightness of SE-Sync using the matrix representation [**7**]. In general, CPL-SLAM and SE-Sync are said to be tight if the globally optimal solution is exactly recovered from the semidefinite relaxation, or equivalently, there is no suboptimality gap between the rounded solution and the relaxed solution.

In our experiments, a City dataset consists of 25×25 square grids in which each grid has side length of 1 m, a robot trajectory of $n = 3000$ poses along the rectilinear path of the grid, odometric measurements that are available between sequential poses along the robot trajectory, and loop-closure measurements that are available at random between non-sequential poses with a probability $p_C = 0.1$. The odometric and loop-closure measurements are generated from noise models of Eqs. (3.10a) and (3.10b), and the default translational weight factor is $\tau_{ij} = 88.89$ that corresponds to an expected translational RMSE of $\sigma_t = 0.15$ m and the default rotational weight factor is $\kappa_{ij} = 40.53$ that corresponds to an expected angular RMSE of $\sigma_R = 0.05\pi$ rad. For the datasets, we vary translational and rotational measurement weight factors τ_{ij} and κ_{ij} individually that correspond to translational and angular RMSEs of $\sigma_t = 0.1 \sim 0.3$ m and $\sigma_R = 0.03\pi \sim 0.15\pi$ rad, respectively, while keeping the other weight factor as the default value.

The results of CPL-SLAM and SE-Sync on the simulated City datasets with high translational and rotational measurement noise are in Figs. 3.3 and 3.4, respectively. For each translational and angular RMSE, we calculate the successful rates of exact recovery from the semidefinite relaxation (Fig. 3.3a and Fig. 3.4a), the relative suboptimality bounds between rounded and relaxed solutions (Fig. 3.3b and Fig. 3.4b), and the objective values of rounded and relaxed solutions (Fig. 3.3c and Fig. 3.4c) statistically from 50 randomly generated City datasets, in which we assume the globally optimal solution is

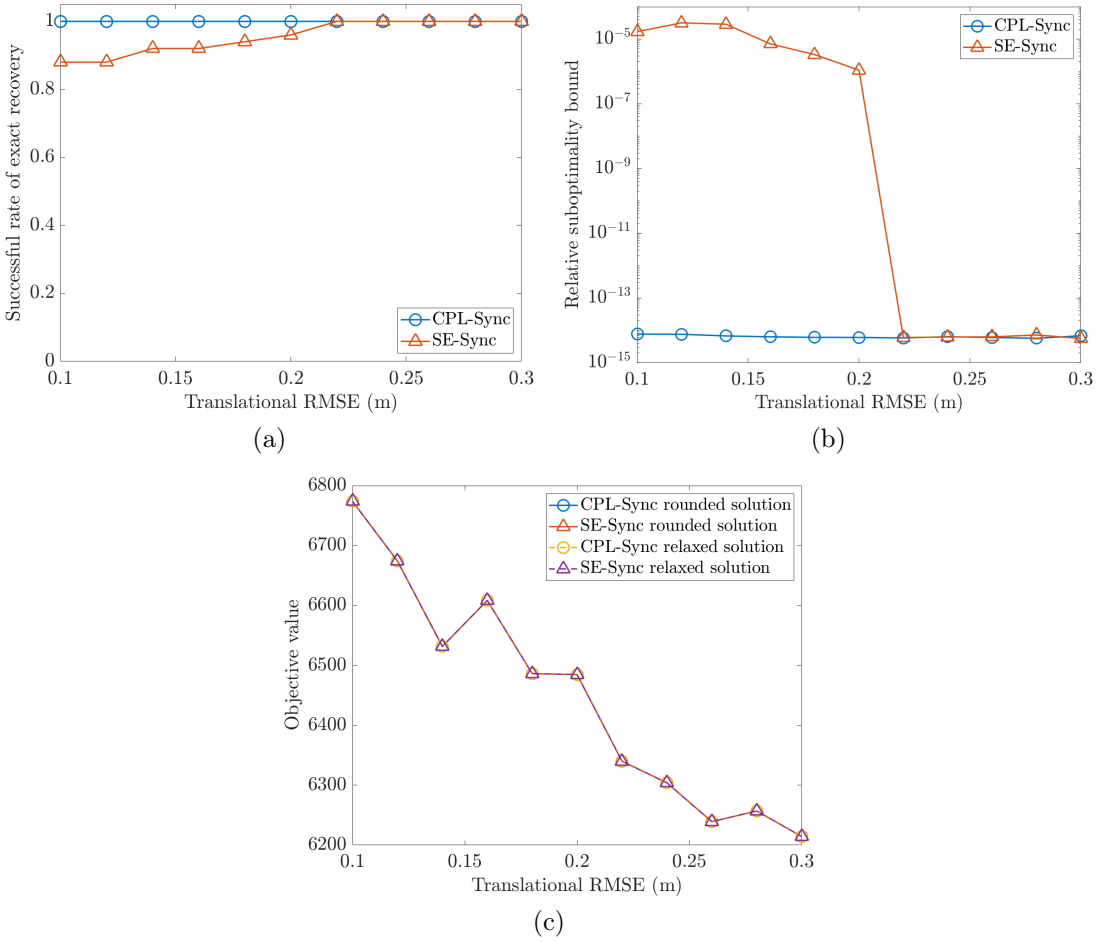


Figure 3.3. The comparisons of CPL-SLAM and SE-Sync on the City datasets with high translational measurement noise with $n = 3000$, $p_C = 0.1$, $\kappa_{ij} = 40.53$ corresponding to angular RSME of $\sigma_R = 0.05\pi$ rad and varying τ_{ij} corresponding to different translational RSMEs of $\sigma_t = 0.1 \sim 0.3$ m. The results are (a) successful rates of exact recovery from the semi-definite relaxation, (b) relative suboptimality bounds between rounded and relaxed solutions, and (c) objective values of rounded and relaxed solutions. For all the datasets with different σ_t, CPL-SLAM has a tighter semidefinite relaxation and is more robust to translational measurement noise.

exactly recovered if the relative suboptimality bound is less than 1×10^{-6}. From Fig. 3.3, it can be seen that CPL-SLAM holds the tightness on all the datasets with translational RMSEs of $\sigma_t = 0.1 \sim 0.3$ m, whereas SE-Sync fails on some of the datasets. From Fig. 3.4, it can be seen that when the angular RMSE is small, i.e., approximately less than 0.15

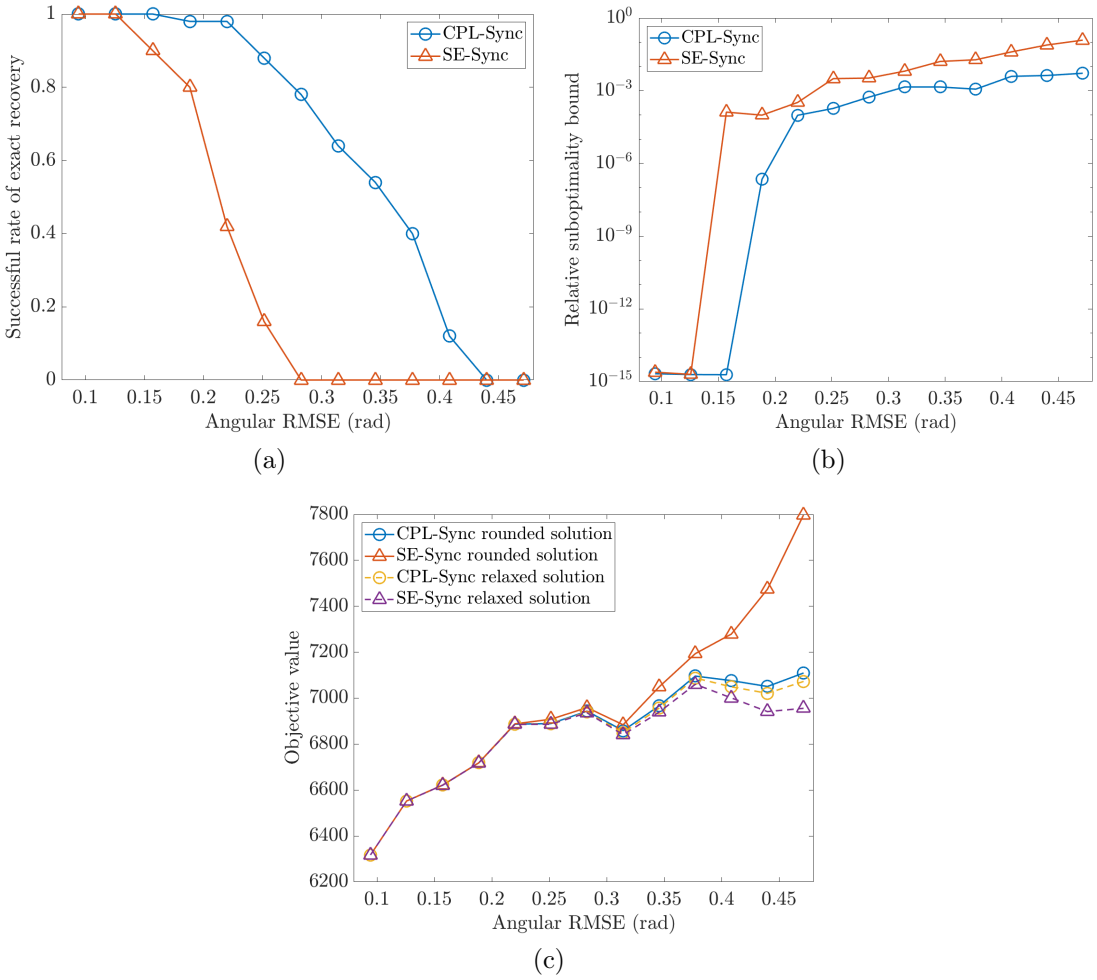


Figure 3.4. The comparisons of CPL-SLAM and SE-Sync on the City datasets with high rotational measurement noise with $n = 3000$, $p_C = 0.1$, $\tau_{ij} = 88.89$ corresponding to translational RSME of $\sigma_t = 0.15$ m and varying κ_{ij} corresponding to different angular RSMEs of $\sigma_R = 0.03\pi \sim 0.15\pi$ rad. The results are (a) successful rates of exact recovery from the semidefinite relaxation, (b) relative suboptimality bounds between rounded and relaxed solutions, and (c) objective values of rounded and relaxed solutions. For all the datasets with different σ_R, CPL-SLAM has a tighter semidefinite relaxation and is more robust to rotational measurement noise.

rad, both CPL-SLAM and SE-Sync exactly recover the globally optimal solution from the semidefinite relaxation, and as angular RMSE increases and is greater than 0.15 rad, CPL-SLAM and SE-Sync begin to fail. In spite of this, we find that CPL-SLAM has a much higher successful rate of exact recovery from the semidefinite relaxation (Fig. 3.3a

and Fig. 3.4a) and orders of magnitude smaller relative suboptimality bounds (Fig. 3.3b and Fig. 3.4b). Furthermore, for the objective value, CPL-SLAM has greater lower bound from the relaxed solution but lower upper bound from the rounded solution in scenarios of high measurement noise (Fig. 3.3c and Fig. 3.4c). All of these results indicate that CPL-SLAM has a tighter semidefinite relaxation using the complex number representation than SE-Sync using the matrix representation, and thus, is more robust to translational and rotational measurement noise.

In Fig. 3.3, it is interesting to see that SE-Sync fails on datasets with small translational measurement noise but works on datasets with large translational measurement noise. Even though there is lack of formal analysis, we guess this is because the tightness of the semidefinite relaxation in SE-Sync, in addition to the magnitude of measurement noise, is also related with the ratio τ_{ij}/κ_{ij} of translational weight factors τ_{ij} and rotational weight factors κ_{ij}, i.e., when τ_{ij}/κ_{ij} increases, the semidefinite relaxation in SE-Sync tends to be relatively more sensitive to measurement nose.

It is obvious that the improved tightness and robustness of CPL-SLAM over SE-Sync in planar graph-based is associated with the more concise representation of complex numbers over matrices in the semidefinite relaxation, for which a theoretically complete analysis similar to [**101**] is left as future work. In spite of this, we present one possible reason that might help explain the improved tightness of CPL-SLAM. The semidefinite matrix resulting from the solution to planar graph-based SLAM using the matrix representation should take the form $X_R = \begin{bmatrix} X_{R_{ij}} \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$ in which each (i,j)-th block $X_{R_{ij}}$ has the algebraic structure $X_{R_{ij}} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \in \mathbb{R}^{2\times 2}$, and SE-Sync drops such an algebraic structure

in the semidefinite relaxation. Even though it is possible for SE-Sync to keep this algebraic structure by either reformulating the associated data matrix or adding numbers of extra linear constraints, substantial computational efforts are required for both options. In comparison, CPL-SLAM preserves the algebraic structure of $X_{R_{ij}}$ as complex numbers in the semidefinite relaxation without having to reformulate the data matrix or introduce any extra constraints. As said before, the explanations given above are still hypotheses and need to be proved. However, we can still conclude from the Figs. 3.3 and 3.4 that the semidefinite relaxation in CPL-SLAM using the complex number representation is tighter than that in SE-Sync using the matrix representation, which further suggests that CPL-SLAM is more robust to measurement noise than SE-Sync.

3.8.3. SLAM Benchmark Datasets

In this subsection, we implement CPL-SLAM, SE-Sync and PDL-GN on a variety of 2D SLAM benchmark datasets with and without landmarks. In these datasets, city10000, M3500, M3500-a, M3500-b, M3500-c and tree10000 are simulated benchmark datasets while the others, i.e., ais2klinik, CSAIL, intel, FR-079, MIT and victoria-park, are real-world datasets. In addition, tree10000 and victoria-park have positions of observed landmarks involved. The chordal initialization [**8**] is used for all the benchmark datasets tested.

For all the 2D SLAM benchmark datasets, CPL-SLAM, SE-Sync and PDL-GN converge to the globally optimal solution. The results are shown in Table 3.1, in which n is the number of unknown poses and n' is the number of observed landmarks, m is the number of pose-pose measurements and m' is the number of pose-landmark measurements, f^* is the globally optimal objective value, and the total time accounts for all the time

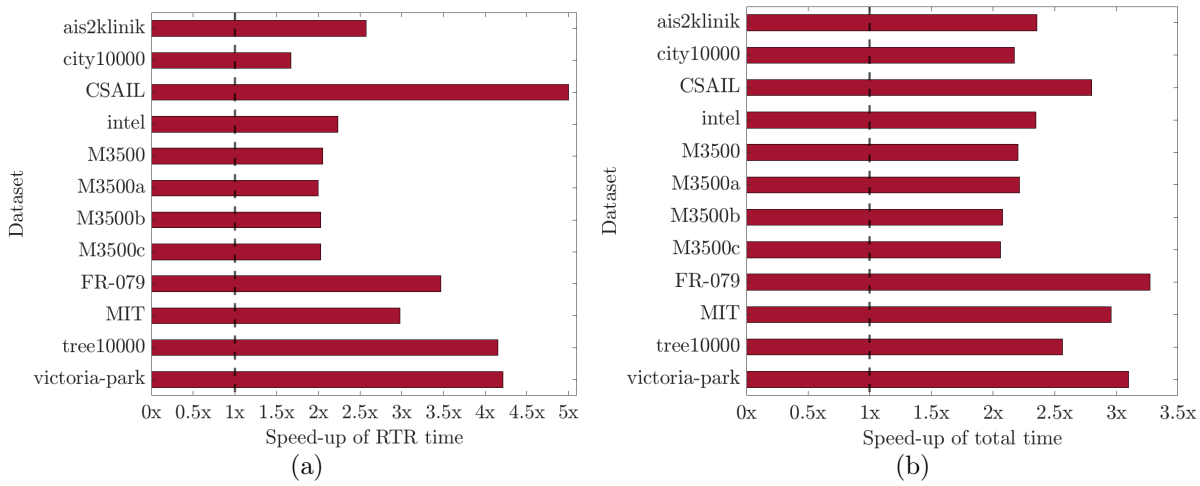


Figure 3.5. The speed-up of CPL-SLAM over SE-Sync on 2D SLAM benchmark datasets. The results are (a) the speed-up of RTR time of CPL-SLAM over SE-Sync and (b) the speed-up of total time of CPL-SLAM over SE-Sync. CPL-SLAM is on average 2.87 and 2.51 times faster than SE-Sync for RTR time and total time, respectively.

taken to solve graph-based SLAM and the RTR time only accounts for the time taken by the RTR method to solve Riemannian staircase optimization. A specific comparison of SE-Sync and CPL-SLAM is further shown in Fig. 3.5. From Table 3.1 and Fig. 3.5, it can be seen that CLP-Sync is significantly faster than both SE-Sync and PDL-GN on all the SLAM benchmark datasets, in which CPL-SLAM outperforms PDL-GN by a factor of 5.53 on average for the overall computation, and outperforms SE-Sync by a factor of 2.87 and 2.51 on average for the computation of the RTR method and the overall computation, respectively. In particular, CPL-SLAM obtains a further improved performance of the RTR method over SE-Sync on the datasets with landmarks, and we think it is due to the conciseness of the complex number representation whose resulting preconditioner accelerates the truncated conjugate gradient method that is used in the RTR method to evaluate the descent direction.

Table 3.1. Results of the 2D SLAM Benchmark Datasets

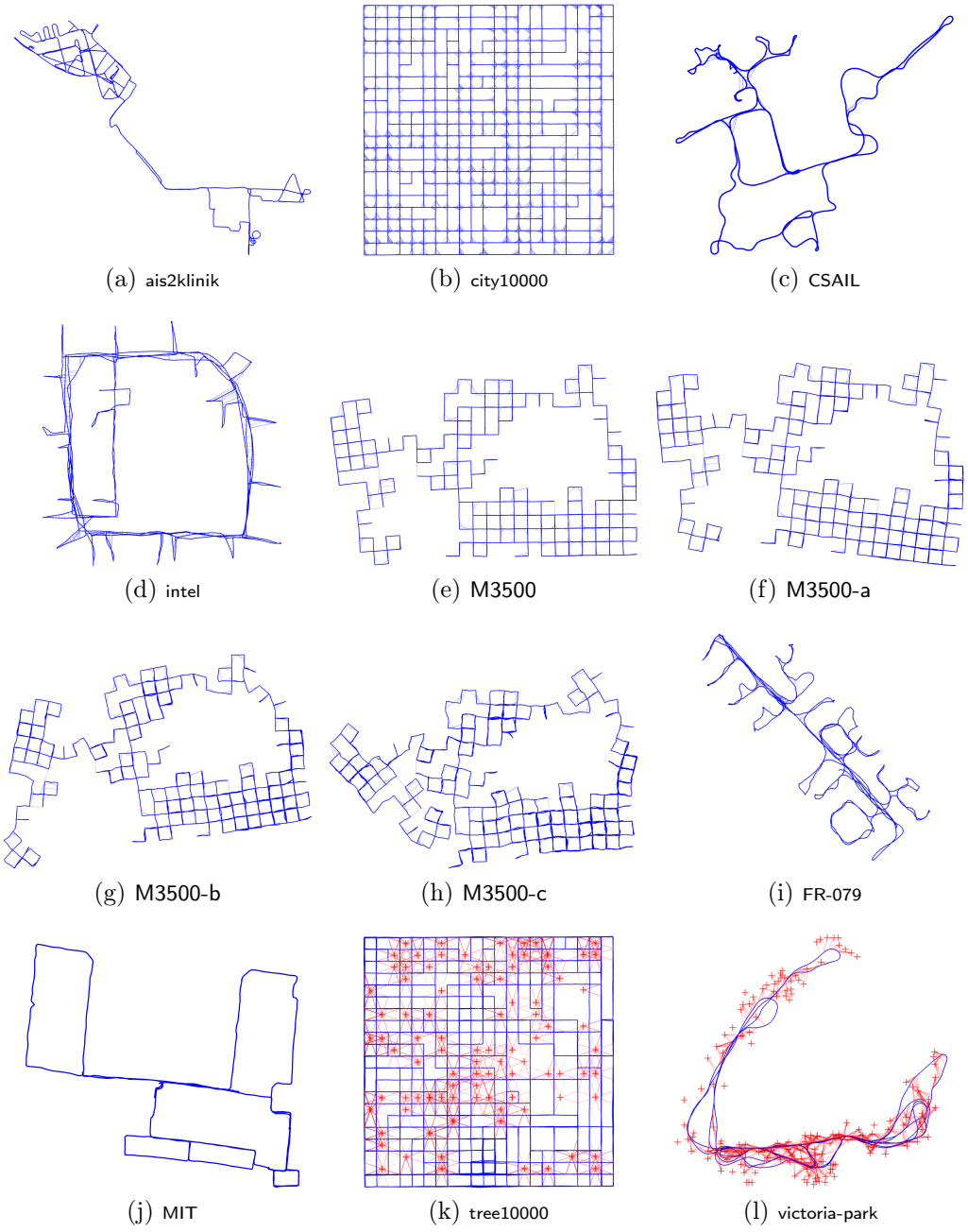
Dataset	$n+n'$	$m+m'$	f^*	PDL-GN [**63**]	SE-Sync [**7**]		CPL-SLAM [ours]	
				Total time (s)	RTR time (s)	Total time (s)	RTR time (s)	Total time (s)
ais2klinik	15115	16727	1.885×10^2	3.2×10^0	2.6×10^0	2.7×10^0	1.0×10^0	1.2×10^0
city10000	10000	20687	6.386×10^2	1.8×10^0	8.6×10^{-1}	1.2×10^0	5.2×10^{-1}	5.4×10^{-1}
CSAIL	1045	1172	3.170×10^1	2.6×10^{-2}	5.0×10^{-3}	1.4×10^{-2}	1.0×10^{-3}	5.0×10^{-3}
intel	1728	2512	5.236×10^1	1.3×10^{-1}	3.8×10^{-2}	6.1×10^{-2}	1.7×10^{-2}	2.6×10^{-2}
M3500	3500	5453	1.939×10^2	3.3×10^{-1}	1.5×10^{-1}	2.2×10^{-1}	7.4×10^{-2}	9.8×10^{-2}
M3500-a	3500	5453	1.598×10^3	4.1×10^{-1}	1.6×10^{-1}	2.3×10^{-1}	8.0×10^{-2}	1.0×10^{-1}
M3500-b	3500	5453	3.676×10^3	1.6×10^0	5.3×10^{-1}	5.9×10^{-1}	2.6×10^{-1}	2.8×10^{-1}
M3500-c	3500	5453	4.574×10^3	2.4×10^0	7.5×10^{-1}	8.2×10^{-1}	3.7×10^{-1}	4.0×10^{-1}
FR-079	989	1217	2.859×10^1	3.9×10^{-2}	5.9×10^{-3}	1.8×10^{-2}	1.7×10^{-3}	5.5×10^{-3}
MIT	808	827	6.115×10^1	7.4×10^{-2}	1.4×10^{-2}	2.1×10^{-2}	4.7×10^{-3}	7.1×10^{-3}
tree10000	10100	14442	6.035×10^2	6.9×10^{-1}	5.4×10^{-1}	5.9×10^{-1}	1.3×10^{-1}	2.3×10^{-1}
victoria-park	7120	10608	4.660×10^2	2.1×10^0	5.9×10^{-1}	6.2×10^{-1}	1.4×10^{-1}	2.0×10^{-1}

The globally optimal results of CPL-SLAM on these 2D SLAM benchmark datasets are as shown in Fig. 3.6. It should be noted that M3500-a, M3500-b and M3500-c in Fig. 3.6f-Fig. 3.6h respectively have extra Gaussian noise with standard deviation 0.1 rad, 0.2 rad and 0.3 rad added to the rotational measurements of M3500 [**72**] in Fig. 3.6e, which indicates that CPL-SLAM can tolerate noisy measurements that are orders of magnitude greater than real-world SLAM applications. For tree10000 in Fig. 3.6k and victoria-park in Fig. 3.6l with landmarks, we denote the positions of landmarks with red "+".

3.9. Conclusion

In this chapter, we have presented CPL-SLAM that is a certifiably correct algorithm for planar graph-based SLAM using the complex number representation. By leveraging the complex semidefinite programming and Riemannian staircase optimization on complex oblique manifolds, CPL-SLAM is applicable to planar graph-based SLAM with and without landmarks. In addition, even though CPL-SLAM essentially solves the complex semidefinite relaxation, we prove that CPL-SLAM exactly retrieves the globally optimal solution to planar graph-based SLAM as long as the noise magnitude is below a certain threshold.

CPL-SLAM is compared with the state-of-the-art methods SE-Sync [**7**] and Powell's Dog-Leg [**63**,**89**] on the simulated Tree datasets, the simulated City datasets and numerous large 2D simulated and real-world SLAM benchmark datasets in terms of scalability and robustness. The results of the data experiments indicate that CPL-SLAM is capable of solving planar graph-based SLAM certifiably, and more importantly, is more efficient in



(a) ais2klinik (b) city10000 (c) CSAIL

(d) intel (e) M3500 (f) M3500-a

(g) M3500-b (h) M3500-c (i) FR-079

(j) MIT (k) tree10000 (l) victoria-park

Figure 3.6. The globally optimal results of CPL-SLAM on 2D SLAM benchmark datasets. Note that CPL-SLAM still obtains global optima on M3500-a, M3500-b and M3500-c in (f)-(g), which respectively has large extra noise with standard deviations of 0.1 rad, 0.2 rad and 0.3 rad added to the rotational measurements of M3500 in (e). For tree10000 in (k) and victoria-park in (l) with landmarks, we denote the positions of landmarks with red "+".

numerical computation and more robust to measurement noise. Thus, we expect that CPL-SLAM outperforms existing state-of-the-art methods to planar graph-based SLAM.

There is still great potential for improvements of CPL-SLAM in several aspects. A fully distributed extension of CPL-SLAM is definitely beneficial to multi-robot simultaneous localization and mapping. In spite of being able to tolerate large measurement noise, CPL-SLAM still needs to enhance its robustness to measurement outliers. At last, it is currently assumed that the positions of landmarks are fully known in CPL-SLAM, and we hope that in the future CPL-SLAM can handle range-only and bearing-only measurements of landmarks, which is another important extension.

3.10. Proofs

In this section, we present proofs of the lemmas and propositions in Section 3.6. These proofs draw heavily on [**90**] and follows a similar procedure to that of [**7**, Appendix C] and [**79**, Section 4.3].

3.10.1. Proof of Lemma 3.1

It is known that the unconstrained Euclidean gradient of $F(z) \triangleq z^H M z$ is $\nabla F(z) = 2Mz$, and thus, if we let $S(z) \triangleq M - \Re\{\mathrm{ddiag}(Mzz^H)\}$, the Riemannian gradient is

$$\begin{aligned} \mathrm{grad}\, F(z) &= \mathrm{proj}_x(\nabla F(z)) \\ &= 2(M - \Re\{\mathrm{ddiag}(Mzz^H)\})z \\ &= 2S(z)z, \end{aligned} \tag{3.32}$$

in which the linear projection operator $\mathrm{proj}_z : \mathbb{C}^n \to T_z\mathbb{C}_1^n$ is defined to be

$$\mathrm{proj}_z u = u - \Re\{\mathrm{ddiag}(uz^H)\}z.$$

In addition, it should be noted that we have assumed that $\mathbb{C}_1^n$ is a Riemannian submanifold of Euclidean space, then the Riemannian Hessian is

$$\mathrm{Hess}\, F(z)[\dot{z}] = \mathrm{proj}_z \mathrm{D}\,\mathrm{grad}\, F(z)[\dot{z}] = \mathrm{proj}_z 2S(z)\dot{z}, \tag{3.33}$$

in which $\mathrm{D}\,\mathrm{grad}\, F(z)[\dot{z}]$ is the direction derivative of $\mathrm{grad}\, F(z)$ along direction $\dot{z}$. From Eq. (3.33), we obtain

$$\big\langle \mathrm{Hess}\, F(z)[\dot{z}], \dot{z} \big\rangle = 2\big\langle S(z)\dot{z}, \dot{z} \big\rangle.$$

Moreover, according to **[90**, Chapter 5], if $\exp_z : T_z\mathbb{C}_1^n \to \mathbb{C}_1^n$ is the exponential map at $z \in \mathbb{C}_1^n$, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t} F \circ \exp_z(t\dot{z})\bigg|_{t=0} = \big\langle \mathrm{grad}\, F(z), \dot{z} \big\rangle$$

and

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} F \circ \exp_z(t\dot{z})\bigg|_{t=0} = \big\langle \mathrm{Hess}\, F(z)[\dot{z}], \dot{z} \big\rangle.$$

Therefore, if $\hat{z} \in \mathbb{C}_1^n$ is a local optimum for Eq. (QP) and $\hat{S} = S(\hat{z})$, it is required that $\hat{S}\hat{z} = 0$ and $\big\langle \dot{z}, \hat{S}\dot{z} \big\rangle \geq 0$ for all $\dot{z} \in T_x\mathbb{C}_1^n$, which completes the proof.

3.10.2. Proof of Lemma 3.2

It should be noted that (1) to (5) in Lemma 3.2 are KKT conditions of (SDP), which proves the necessity. Since the identity matrix $\mathbf{I} \in \mathbb{C}^{n\times n}$ is strictly feasible to Lemma 3.2, the Slater's condition is satisfied, which proves the sufficiency. In addition, it should be

noted that the Slater's condition also holds for the dual of (SDP). If $\text{rank}(\hat{S}) = n - 1$, according to [**102**, Theorem 6], $\hat{S}$ is dual nondegenerate. Moreover, by complementary slackness, $\hat{S}$ is also optimal for the dual of (SDP), which, as a result of [**102**, Theorem 10], implies that $\hat{X}$ is unique. If $\text{rank}(\hat{S}) = n - 1$, it can be concluded that $\hat{X}$ has rank one from $\hat{S}\hat{X} = \mathbf{0}$.

3.10.3. Proof of Lemma 3.3

Since $\hat{z} \in \mathbb{C}_1^n$ is a first-order critical point and $\hat{S} \succeq 0$, we conclude that $\hat{z}$ is a second-order critical point from Lemma 3.1. Also it can be checked that $\hat{X} = \hat{z}^H \hat{z} \in \mathbb{H}^n$ satisfies (1) to (5) in Lemma 3.2, thus, $\hat{z}$ solves (QP), and $\hat{X}$ solves (SDP) and is the unique global optimum for (SDP) if $\text{rank}(\hat{S}) = n - 1$.

3.10.4. Proof of Proposition 3.6.1

In order to prove Proposition 3.6.1, we need Propositions 3.10.1 and 3.10.2 as follows.

Proposition 3.10.1. If $\underline{M} \in \mathbb{H}^n$ is data matrix of the form Eq. (3.18) that is constructed with the true (latent) relative measurements, and $\underline{z} \in \mathbb{C}_1^n$ is the true (latent) value of rotational states z, then $\underline{M}\,\underline{z} = \mathbf{0}$ and $\lambda_2(\underline{M}) > 0$.

Proof. For consistency, we assume that (P) and (QP) are formulated with the true (latent) relative measurements. Let $\underline{s} \in \mathbb{C}^{n'}$ and $\underline{c} \in \mathbb{C}^n$ be the true (latent) value of landmark positions and translational states c, respectively, then $\underline{\xi} = \begin{bmatrix} \underline{s}^\top & \underline{c}^\top & \underline{z}^\top \end{bmatrix}^\top \in \mathbb{C}^{n'} \times \mathbb{C}^n \times \mathbb{C}_1^n$ solves (P), and the optimal objective value is 0. Since (QP) is equivalent to (P), it can be concluded that $\underline{z} \in \mathbb{C}_1^n$ solves (QP), and the optimal objective value of (QP)

is 0 as well. Furthermore, since $\underline{M} \succeq 0$, we obtain $\underline{M}\,\underline{z} = \mathbf{0}$. Let $\Xi \triangleq \text{diag}\{\underline{z}_1, \cdots, \underline{z}_n\} \in \mathbb{C}^{n\times n}$ and $L(W^z) \in \mathbb{R}^{n\times n}$ be the Laplacian such that

$$[L(W^z)]_{ij} \triangleq \begin{cases} \sum\limits_{(i,k)\in\mathcal{E}} \kappa_{ik}, & i = j, \\ -\kappa_{ij}, & (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise}, \end{cases}$$

we obtain $L(\underline{G}^z) = \Xi L(W^z)\Xi^H$. It should be noted that G is assumed to be connected, and as a result, $\lambda_2(L(\underline{G}^z)) > 0$ and $L(\underline{G}^z)\underline{z} = \mathbf{0}$. Furthermore, it is by the definition of $\underline{M}$ or M in Eq. (3.14) that

$$\underline{M} = L(\underline{G}^z) + \underline{M}^\sigma, \tag{3.34}$$

in which $\underline{M}^\sigma = \underline{\Sigma}^z - \underline{\Theta}^H \Lambda^\dagger \underline{\Theta}$. From Eqs. (3.21a) to (3.21c) and (3.22), we obtain that $\underline{M}^\sigma$ is the Schur complement of

$$\begin{bmatrix} B_1^H B_1 & B_1^H B_2 \\ B_2^H B_1 & B_2^H B_2 \end{bmatrix} = \begin{bmatrix} B_1^H \\ B_2^H \end{bmatrix} \begin{bmatrix} B_1 & B_2 \end{bmatrix} \succeq 0,$$

which suggests that $\underline{M}^\sigma \succeq 0$ and $\lambda_1(\underline{M}^\sigma) \geq 0$. As a result of Eq. (3.34), $\lambda_2(L(\underline{G}^z)) > 0$ and $\lambda_1(\underline{M}^\sigma) \geq 0$, we obtain

$$\lambda_2(\underline{M}) \geq \lambda_2(L(\underline{G}^z)) + \lambda_1(\underline{M}^\sigma) > 0,$$

which completes the proof. □

Proposition 3.10.2. If $\underline{z} \in \mathbb{C}_1^n$ is the true (latent) value of $z \in \mathbb{C}_1^n$, and $\hat{z}$ solves (QP), and $d(\underline{z}, \hat{z}) \triangleq \min_{\theta \in \mathbb{R}} \|\hat{z} - e^{\mathbf{i}\theta}\underline{z}\|$, then we obtain

$$d(\underline{z}, \hat{z}) \leq 2\sqrt{\frac{n\|M - \underline{M}\|_2}{\lambda_2(\underline{M})}} \tag{3.35}$$

Proof. If we define $\Delta M \triangleq M - \underline{M} \in \mathbb{H}^n$ to be the perturbation matrix, then

$$\underline{z}^H M \underline{z} = \underline{z}^H \underline{M}\, \underline{z} + \underline{z}^H \Delta M \underline{z} = \underline{z}^H \Delta M \underline{z} \leq n\|\Delta M\|_2, \tag{3.36}$$

in which, according to Proposition 3.10.1, $\underline{z}^H \underline{M}\, \underline{z} = 0$. In addition, it should be noted that

$$\underline{z}^H M \underline{z} \geq \hat{z}^H M \hat{z} \tag{3.37}$$

and

$$\hat{z}^H M \hat{z} = \hat{z}^H \underline{M} \hat{z} + \hat{z}^H \Delta M \hat{z} \geq \hat{z}^H \underline{M} \hat{z} - n\|\Delta M\|_2. \tag{3.38}$$

From Eqs. (3.36) to (3.38), we obtain

$$2n\|\Delta M\|_2 \geq \hat{z}^H \underline{M} \hat{z} \tag{3.39}$$

From Proposition 3.10.1, we obtain

$$\begin{aligned} \hat{z}^H \underline{M} \hat{z} &= (\hat{z} - \frac{1}{n}\underline{z}^H \hat{z} \underline{z})^H \underline{M} (\hat{z} - \frac{1}{n}\underline{z}^H \hat{z} \underline{z}) \\ &\geq \lambda_2(\underline{M}) \|\hat{z} - \frac{1}{n}\underline{z}^H \hat{z} \underline{z}\|^2, \end{aligned} \tag{3.40}$$

in which the equality "=" uses $\underline{M}\,\underline{z} = \mathbf{0}$ and the inequality "$\geq$" uses $\lambda_2(\underline{M}) > \lambda_1(\underline{M}) = 0$ and $\underline{z}^H(\hat{z} - \frac{1}{n}\underline{z}^H\hat{z}\underline{z}) = 0$. Furthermore, an algebraic manipulation indicates that

$$\|\hat{z} - \frac{1}{n}\underline{z}^H\hat{z}\underline{z}\|^2 = n - \frac{1}{n}|\underline{z}^H\hat{z}|^2. \tag{3.41}$$

From Eqs. (3.40) and (3.41), we obtain

$$\begin{aligned} \hat{z}^H\underline{M}\hat{z} &\geq \lambda_2(\underline{M})\|\hat{z} - \frac{1}{n}\underline{z}^H\hat{z}\underline{z}\|^2 \\ &= \frac{1}{n}\lambda_2(\underline{M})(n^2 - |\underline{z}^H\hat{z}|^2) \\ &\geq \lambda_2(\underline{M})(n - |\underline{z}^H\hat{z}|), \end{aligned} \tag{3.42}$$

in which the last inequality "$\geq$" uses the Cauchy-Schwarz inequality

$$|\underline{z}^H\hat{z}| \leq ||\underline{z}|| \cdot \|\hat{z}\| = n.$$

Substituting Eq. (3.42) into Eq. (3.39) and simplifying the resulting equation, we obtain

$$n - |\underline{z}^H\hat{z}| \leq \frac{2n\|\Delta M\|_2}{\lambda_2(\underline{M})}. \tag{3.43}$$

In addition, from [**79**, Eq. (4.1)], it is known that $d(\underline{z},\, \hat{z}) = \sqrt{2n - 2|\underline{z}^H\hat{z}|}$, and then from Eq. (3.43), we further obtain Eq. (3.35), which completes the proof. □

To prove Proposition 3.6.1, we first decompose $\hat{S} = M - \Re(\mathrm{ddiag}(M\hat{z}^H\hat{z}))$ as follows:

$$\begin{aligned}
\hat{S} =& M - \Re(\mathrm{ddiag}(M\hat{z}^H\hat{z})) \\
=& \underline{M} + \Delta M - \\
& \qquad \Re\left\{\mathrm{ddiag}\left((\underline{M}+\Delta M)(\underline{z}+\Delta z)(\underline{z}+\Delta z)^H\right)\right\} \\
=& \underline{M} + \Delta M - \Re\big\{\mathrm{ddiag}(\underline{M}\Delta z z^H + \underline{M}\Delta z \Delta z^H + \\
& \qquad \underbrace{\Delta M(\underline{z}+\Delta z)(\underline{z}+\Delta z)^H)\big\}}_{\Delta S},
\end{aligned}$$

in which $\underline{z} \in \mathbb{C}_1^n$ is the true (latent) value of $z \in \mathbb{C}_1^n$ such that $\underline{M}\,\underline{z} = \mathbf{0}$, and $\hat{z}$ solves (QP), and $\Delta z \triangleq \hat{z} - \underline{z}$. In addition, we assume $\|\hat{z} - \underline{z}\| = d(\hat{z}, \underline{z}) \triangleq \min\limits_{\theta\in\mathbb{R}} \|\hat{z} - e^{\mathbf{i}\theta}\underline{z}\|$. It is obvious that $\|\Delta S\|_2 \to 0$ as long as $\|\Delta M\|_2 \to 0$ and $\|\Delta z\| \to 0$, and by Proposition 3.10.2, we obtain $\|\Delta z\| \to 0$ as long as $\|\Delta M\|_2 \to 0$. As a result, from continuity, there exists some $\gamma > 0$ such that $\|\Delta S\|_2 < \lambda_2(\underline{M})$ as long as $\|\Delta M\|_2 < \gamma$. Then we obtain

$$\lambda_i(\hat{S}) \geq \lambda_i(\underline{M}) - \|\Delta S\|_2 > \lambda_i(\underline{M}) - \lambda_2(\underline{M}) \geq 0$$

for all $i \geq 2$, which implies that $\hat{S}$ at least has $n-1$ positive eigenvalues. In addition, by Lemma 3.1, we obtain $\hat{S}\hat{z} = \mathbf{0}$, from which it can be concluded that $\hat{S} \succeq 0$ and $\mathrm{rank}(\hat{S}) = n-1$. Furthermore, Lemma 3.3 guarantees that $\hat{X} = \hat{z}\hat{z}^H \in \mathbb{H}^n$ is the unique optimum of (SDP) if $\hat{S} \succeq 0$ and $\mathrm{rank}(\hat{S}) = n-1$.

CHAPTER 4

Majorization Minimization Methods for Distributed Pose Graph Optimization

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

4.1. Introduction

Pose graph optimization (PGO) is a nonlinear and nonconvex optimization problem estimating unknown poses from noisy relative pose measurements. PGO associates each pose with a vertex and each relative pose measurement with an edge, from which the optimization problem is well represented through a graph. PGO has important applications in a number of areas, including but not limited to robotics [**16**, **18**, **45**], autonomous driving [**47**], and computational biology [**77**, **78**]. Recent advances [**7**, **63**, **72**, **73**, **75**, **87**–**89**, **103**, **104**] suggest that PGO can be well solved using iterative optimization. However, the aforementioned techniques [**7**, **63**, **72**, **73**, **75**, **87**–**89**, **103**, **104**] rely on a centralized optimizer to solve PGO and are difficult to distribute across a network. Due to communication and computational limitations, most, if not all, of these techniques [**7**, **63**, **72**, **73**, **75**, **87**–**89**, **103**, **104**] are only applicable to small- and medium-sized problems. Moreover, their centralized pipelines are equivalent to using a master node to aggregate information from the entire network, making it impossible to meet potential privacy requirements[1] one may wish to impose [**105**, **106**].

In multi-robot simultaneous localization and mapping (SLAM) [**107**–**116**], each robot estimates not only its own poses but those of the others as well to build an environment map. Even though such a problem can be solved by PGO, communication between robots is restricted and multi-robot SLAM has more unknown poses than single-robot SLAM. Thus, instead of using centralized PGO [**7**, **63**, **72**, **73**, **75**, **87**–**89**, **103**, **104**], it is more reasonable to formulate this large-sized estimation problem involving multiple robots as distributed PGO—each robot in multi-robot SLAM is represented as a node and two nodes

[1]In terms of "privacy", we mean that only peer-to-peer communication between neighboring nodes is required.

(robots) are said to be neighbors if there exists a noisy relative pose measurement between them (a more detailed description of distributed PGO can be found in Section 4.4). In most cases, it is assumed that inter-node communication only occurs between neighboring nodes and most of these iterative optimization methods [**7**, **63**, **72**, **73**, **75**, **87**–**89**, **103**, **104**] are infeasible, which renders distributed PGO more challenging than centralized PGO.

In this chapter, we propose majorization minimization (MM) methods [**117**, **118**] for distributed PGO. As the name would suggest, MM methods have two steps. First, in the majorization step, we construct a surrogate function that majorizes the objective function, i.e., the surrogate function is an upper bound of the objective function except for the current iterate at which both functions attain the same value. Then, in the minimization step, we minimize the surrogate function instead of the original objective function to improve the current iterate. Even though the procedure is straightforward, MM methods remain difficult for practical use—a suitable surrogate function, whose construction and minimization can not be more difficult than solving the optimization problem itself, is not generally evident, and MM methods might converge to noncritical stationary points for nonconvex optimization problems and suffer from slow convergence around stationary points. The implementation of MM methods on large-scale, complicated and nonconvex optimization problems like distributed PGO is nontrivial, and inter-node communication requirements impose extra restrictions making it more so. All of these issues are addressed in our MM methods for distributed PGO both theoretically and empirically.

The preliminary results of this chapter have been presented in [**1**, **59**, **119**]. In particular, we introduced and elaborated on the use of Nesterov's method [**120**, **121**] and adaptive restart [**122**] for the first time to accelerate the convergence of PGO without

sacrificing the theoretical guarantees. Our MM methods in this chapter are also capable of handling a broad class of robust loss kernels, no longer require each iteration to attain a local optimal solution to the surrogate function for the convergence guarantees, and adopt a novel adaptive restart scheme for distributed PGO without a master node to make full use of Nesterov's acceleration.

In summary, the contributions of this chapter are the following:

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

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

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

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

The rest of this chapter is organized as follows. Section 4.2 reviews the state-of-the-art methods for distributed PGO. Section 4.3 introduces mathematical notation and preliminaries that are used in this chapter. Section 4.4 formulates the problem of distributed PGO. Sections 4.5 and 4.6 present surrogate functions for individual loss terms and the

overall distributed PGO, respectively, which are fundamental to our MM methods. Sections 4.7 to 4.9 present unaccelerated and accelerated MM methods for distributed PGO that are guaranteed to converge to first-order critical points, which are the major contributions of this chapter. Section 4.10 implements our MM methods for distributed PGO on a number of simulated and real-world SLAM datasets and make extensive comparisons against existing state-of-the-art methods [**5**, **6**]. Section 4.11 concludes this chapter and discusses future work. Section 4.12 completes the proofs of the propositions presented in this chapter.

4.2. Related Work

In the last decade, multi-robot SLAM has been becoming increasingly popular, which promotes the research of distributed PGO [**5**, **6**, **123**, **124**].

Choudhary et al. [**5**] present a two-stage algorithm that implements either Jacobi Over-Relaxation or Successive Over-Relaxation as distributed linear system solvers. Similar to centralized methods, [**5**] first evaluates the chordal initialization [**8**] and then improves the initial guess with a single Gauss-Newton step. However, one step of Gauss-Newton method in most cases can not lead to sufficient convergence for distributed PGO. In addition, no line search is performed in [**5**] due to the communication limitation, and thus, the behaviors of the single Gauss-Newton step is totally unpredictable and might result in bad solutions.

Tian et al. [**6**] present the distributed certifiably correct PGO using Riemannian block coordinate descent method, which is later generalized to asynchronous and parallel distributed PGO [**125**]. Specially, their method makes use of Riemannian staircase optimization to solve the semidefinite relaxation of distributed PGO and is guaranteed to

converge to global optimal solutions under moderate measurement noise. Following our previous works [**1**, **59**], they implement Nesterov's method for acceleration as well. Contrary to our MM methods, a major drawback of [**6**] is that their method has to precompute red-black coloring assignment for block aggregation and keep part of the blocks in idle for estimate updates. In addition, although several strategies for block selection (e.g., greedy/importance sampling) and Nesterov's acceleration (e.g., adaptive/fixed restarts) are adopted in [**6**] to improve the convergence, most of them are either inapplicable without a master node or at the sacrifice of computational efficiency and theoretical guarantees. In contrast, our MM methods are much faster (see Section 4.10) but have no such restrictions for acceleration. More recently, Tian et al. further apply Riemannian block coordinate descent method to distributed PGO with robust loss kernels [**116**]. However, they solve robust distributed PGO by trivially updating the weights using graduated non-convexity [**126**] and no formal proofs of convergence are provided. Again, this is contrast to the work presented here that has provable convergence to first-order critical points for a broad class of robust loss kernels.

Tron and Vidal [**123**] present a consensus-based method for distributed PGO using Riemannian gradient. The authors derive a condition for convergence guarantees related with the stepsize of the method and the degree of the pose graph. Nonetheless, their method estimates rotation and translation separately, fails to handle robust loss kernels, and needs extra computation to find the convergence-guaranteed stepsize.

Cristofalo et al. [**124**] present a novel distributed PGO method using Lyapunov theory and multi-agent consensus. Their method is guaranteed to converge if the pose graph has certain topological structures. However, [**124**] updates rotations without exploiting the

translational measurements and only applies to pairwise consistent PGO with nonrobust loss kernels.

In comparison to these aforementioned techniques, our MM methods have the mildest conditions (not requiring any specific pose graph structures, any extra computation for preprocessing, any master nodes for information aggregation, etc.) to converge to first-order critical points, apply to a broad class of robust loss kernels in robotics and computer vision, and manage to implement decentralized acceleration with convergence guarantees. Most importantly, as is shown in Section 4.10, our MM methods outperform existing state-of-the-art methods in terms of both efficiency and accuracy on a variety of simulated and real-world SLAM benchmark datasets.

4.3. Notation

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

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

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

$$\left\langle X, Y \right\rangle_M \triangleq \operatorname{trace}(XMY^\top)$$

where X, $Y \in \mathbb{R}^{m\times n}$, and if M is the identity matrix, $\left\langle \cdot, \cdot \right\rangle_M$ might also be denoted as $\left\langle \cdot, \cdot \right\rangle : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}$ such that

$$\left\langle X, Y \right\rangle \triangleq \operatorname{trace}(XY^\top).$$

Norms. The notation $\| \cdot \|$ denotes the Frobenius norm of matrices and vectors. The notation $\| \cdot \|_2$ denotes the induced 2-norms of matrices and linear operators. For a positive semidefinite matrix $M \in \mathbb{R}^{n\times n}$, $\| \cdot \|_M : \mathbb{R}^{m\times n} \to \mathbb{R}^+$ denotes the function

$$\|X\|_M \triangleq \sqrt{\operatorname{trace}(XMX^\top)}$$

where $X \in \mathbb{R}^{m\times n}$.

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

Graph Theory. Let $\overrightarrow{\mathcal{G}} = (\mathcal{V}, \overrightarrow{\mathcal{E}})$ be a directed graph whose vertices are ordered pairs. For any vertices (α, i) and $(\beta, j) \in \mathcal{V}$, the notation $\overrightarrow{\mathcal{E}}^{\alpha\beta}$ denotes the set

$$\overrightarrow{\mathcal{E}}^{\alpha\beta} \triangleq \{(i, j) | ((\alpha, i), (\beta, j)) \in \overrightarrow{\mathcal{E}}\}, \tag{4.1}$$

and the notation $\mathcal{N}_-^\alpha$ denotes the set

$$\mathcal{N}_-^\alpha \triangleq \{\beta | \overrightarrow{\mathcal{E}}^{\alpha\beta} \neq \emptyset \text{ and } \alpha \neq \beta\}, \tag{4.2}$$

and the notation $\mathcal{N}_+^\alpha$ denotes the set

$$\mathcal{N}_+^\alpha \triangleq \{\beta | \overrightarrow{\mathcal{E}}^{\beta\alpha} \neq \emptyset \text{ and } \alpha \neq \beta\}, \tag{4.3}$$

and the notation $\mathcal{N}^\alpha$ denotes the set

$$\mathcal{N}^\alpha \triangleq \mathcal{N}_-^\alpha \cup \mathcal{N}_+^\alpha. \tag{4.4}$$

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

4.4. Problem Formulation

4.4.1. Distributed Pose Graph Optimization

In distributed PGO [**1**,**5**,**6**,**123**], we are given $|\mathcal{A}|$ nodes $\mathcal{A} \triangleq \{1, 2, \cdots, |\mathcal{A}|\}$ and each node $\alpha \in \mathcal{A}$ has n_α poses $g_1^\alpha, g_2^\alpha, \cdots, g_{n_\alpha}^\alpha \in SE(d)$. Let $g_{(\cdot)}^\alpha \triangleq (t_{(\cdot)}^\alpha, R_{(\cdot)}^\alpha)$ where $t_{(\cdot)}^\alpha \in \mathbb{R}^d$ is the translation and $R_{(\cdot)}^\alpha \in SO(d)$ the rotation. We consider the problem of estimating unknown poses $g_1^\alpha, g_2^\alpha, \cdots, g_{n_\alpha}^\alpha \in SE(d)$ for all the nodes $\alpha \in \mathcal{A}$ given intra-node noisy measurements $\tilde{g}_{ij}^{\alpha\alpha} \triangleq (\tilde{t}_{ij}^{\alpha\alpha}, \widetilde{R}_{ij}^{\alpha\alpha}) \in SE(d)$ of the relative pose

$$g_{ij}^{\alpha\alpha} \triangleq \left(g_i^\alpha\right)^{-1} g_j^\alpha \in SE(d) \tag{4.5}$$

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

$$g_{ij}^{\alpha\beta} \triangleq \left(g_i^\alpha\right)^{-1} g_j^\beta \in SE(d) \tag{4.6}$$

between different nodes $\alpha \neq \beta$. In Eqs. (4.5) and (4.6), note that $\tilde{t}_{ij}^{\alpha\alpha}$ and $\tilde{t}_{ij}^{\alpha\beta} \in \mathbb{R}^d$ are translational measurements, and $\widetilde{R}_{ij}^{\alpha\alpha}$ and $\widetilde{R}_{ij}^{\alpha\beta} \in SO(d)$ are rotational measurements.

Following [**7**], we model distributed PGO as a directed graph $\overrightarrow{\mathcal{G}} \triangleq (\mathcal{V}, \overrightarrow{\mathcal{E}})$ whose vertices are ordered pairs consisting of node index, e.g., α and β and pose index, e.g., i and j. In the directed graph $\overrightarrow{\mathcal{G}}$, the vertex $(\alpha,\, i) \in \mathcal{V}$ is in one-to-one correspondence with the unknown pose $g_i^\alpha \in SE(d)$ and the directed edge $((\alpha,\, i),\, (\beta,\, j)) \in \overrightarrow{\mathcal{E}}$ is in one-to-one correspondence with the noisy measurement $\tilde{g}_{ij}^{\alpha\beta} \in SE(d)$. Note that $\overrightarrow{\mathcal{E}}^{\alpha\beta}$, $\mathcal{N}_-^\alpha$, $\mathcal{N}_+^\alpha$ and $\mathcal{N}^\alpha$ in Eqs. (4.1) to (4.4) are well defined for distributed PGO.

From the convention of distributed PGO, nodes α and $\beta \in \mathcal{A}$ are referred as neighbors as long as either $\overrightarrow{\mathcal{E}}^{\alpha\beta} \neq \emptyset$ or $\overrightarrow{\mathcal{E}}^{\beta\alpha} \neq \emptyset$. We remark that $\mathcal{N}^\alpha$ is the set of neighbors that has a directed edge connected with node α, and $\mathcal{N}_-^\alpha$ and $\mathcal{N}_+^\alpha$ are the sets of neighbors that have a directed edge from and to node α, respectively.

In the rest of this paper, we make the following assumption that each node can communicate with its neighbors and the network topology is unchanged during optimization. These assumptions are common in distributed PGO [**5**, **6**, **123**, **124**].

Assumption 4.1. Each node α can communicate with its neighbors $\beta \in \mathcal{N}^\alpha$ and the network topology is unchanged.

4.4.2. Loss Kernels

In practice, it is inevitable that there exist inter-node measurements that are outliers resulting from false loop closures; these adversely affect the overall performance of

distributed PGO. To address this issue, it is popular to use non-trivial loss kernels—e.g., Huber and Welsch losses—to enhance the robustness of distributed PGO against outliers [**127**–**129**].

In this chapter, we make the following assumption that applies to a broad class of loss kernels $\rho(\cdot) : \mathbb{R}^+ \to \mathbb{R}$ in robotics and computer vision.

Assumption 4.2. The loss kernel $\rho(\cdot) : \mathbb{R}^+ \to \mathbb{R}$ satisfies the following properties:

(a) $\rho(s) \geq 0$ for any $s \in \mathbb{R}^+$ and the equality "=" holds if and only if $s = 0$;

(b) $\rho(\cdot) : \mathbb{R}^+ \to \mathbb{R}$ is continuously differentiable for any $s \in \mathbb{R}^+$;

(c) $\rho(\cdot) : \mathbb{R}^+ \to \mathbb{R}$ is a concave function;

(d) $0 \leq \nabla\rho(s) \leq 1$ for any $s \in \mathbb{R}^+$ and $\nabla\rho(0) = 1$;

(e) $\varphi(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}$ with $\varphi(X) \triangleq \rho(\|X\|^2)$ has Lipschitz continuous gradient, i.e., there exists $\mu > 0$ such that $\|\nabla\varphi(X) - \nabla\varphi(X')\| \leq \mu \cdot \|X - X'\|$ for any X, $X' \in \mathbb{R}^{m\times n}$.

In the following, we present some examples of loss kernels (see Fig. 4.1) satisfying Assumption 4.2.

Example 4.1 (Trivial Loss)**.**

$$\rho(s) = s. \tag{4.7}$$

Example 4.2 (Huber Loss)**.**

$$\rho(s) = \begin{cases} s, & |s| \leq a, \\ 2\sqrt{a|s|} - a, & |s| \geq a \end{cases} \tag{4.8}$$

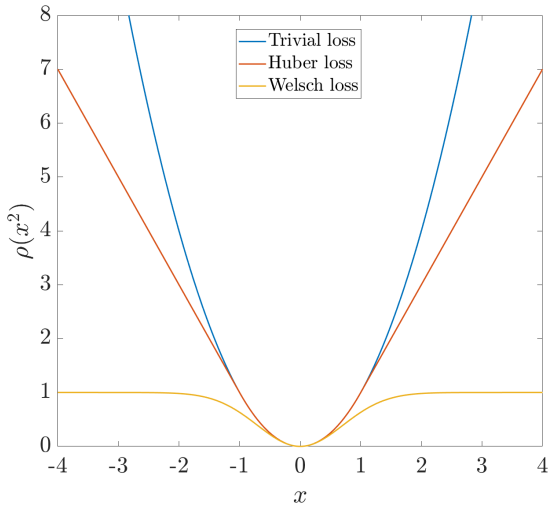


Figure 4.1. $\rho(x^2)$ for trivial, Huber, Welsch losses.

where $a > 0$.

Example 4.3 (Welsch Loss)**.**

$$\rho(s) = a - a \exp\left(-\frac{s}{a}\right) \tag{4.9}$$

where $a > 0$.

4.4.3. Objective Function

Recall that each node $\alpha \in \mathcal{A}$ has n_α unknown poses g_1^α, g_2^α, $\cdots$, $g_{n_\alpha}^\alpha \in SE(d)$. For notational simplicity, we define $\mathcal{X}^\alpha$ and $\mathcal{X}$ as

$$\mathcal{X}^\alpha \triangleq \mathbb{R}^{d \times n_\alpha} \times SO(d)^{n_\alpha}$$

and

$$\mathcal{X} \triangleq \mathcal{X}^1 \times \cdots \times \mathcal{X}^{|\mathcal{A}|} \subset \mathbb{R}^{d \times (d+1)n},$$

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

$$X_i^\alpha \triangleq \begin{bmatrix} t_i^\alpha & R_i^\alpha \end{bmatrix} \in SE(d) \subset \mathbb{R}^{d\times(d+1)},$$

represent $(g_1^\alpha, g_2^\alpha, \cdots, g_{n_\alpha}^\alpha) \in SE(d)^{n_\alpha}$, i.e., all the poses of node $\alpha \in \mathcal{A}$, as an element of $\mathcal{X}^\alpha$ as well as a $d \times (d+1)n_\alpha$ matrix

$$X^\alpha \triangleq \begin{bmatrix} t^\alpha & R^\alpha \end{bmatrix} \in \mathcal{X}^\alpha \subset \mathbb{R}^{d\times(d+1)n_\alpha},$$

where

$$t^\alpha \triangleq \begin{bmatrix} t_1^\alpha & \cdots & t_{n_\alpha}^\alpha \end{bmatrix} \in \mathbb{R}^{d\times n_\alpha}$$

and

$$R^\alpha \triangleq \begin{bmatrix} R_1^\alpha & \cdots & R_{n_\alpha}^\alpha \end{bmatrix} \in SO(d)^{n_\alpha} \subset \mathbb{R}^{d\times dn_\alpha},$$

and represent $\{(g_1^\alpha, g_2^\alpha, \cdots, g_{n_\alpha}^\alpha)\}_{\alpha\in\mathcal{A}} \in SE(d)^n$, i.e., all the poses of distributed PGO, as an element of $\mathcal{X}$ as well as a $d \times (d+1)n$ matrix

$$X \triangleq \begin{bmatrix} X^1 & \cdots & X^{|\mathcal{A}|} \end{bmatrix} \in \mathcal{X} \subset \mathbb{R}^{d\times(d+1)n}.$$

Remark 4.1. $\mathcal{X}^\alpha$ and $\mathcal{X}$ are by definition homeomorphic to $SE(d)^{n_\alpha}$ and $SE(d)^n$, respectively. Thus, $X^\alpha \in \mathcal{X}^\alpha$ and $X \in \mathcal{X}$ are sufficient to represent elements of $SE(d)^{n_\alpha}$ and $SE(d)^n$.

Following [**1**, **7**, **59**], distributed PGO can be formulated as an optimization problem on $X = \begin{bmatrix} X^1 & \cdots & X^{|\mathcal{A}|} \end{bmatrix} \in \mathcal{X}$:

Problem 4.1 (Distributed Pose Graph Optimization)**.**

$$\min_{X\in\mathcal{X}} F(X). \tag{4.10}$$

The objective function $F(X)$ in Eq. (4.10) is defined as

$$\begin{aligned} F(X) \triangleq & \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} \frac{1}{2}\Big[\kappa_{ij}^{\alpha\alpha}\|R_i^\alpha \widetilde{R}_{ij}^{\alpha\alpha} - R_j^\alpha\|^2 + \tau_{ij}^{\alpha\alpha}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\alpha} + t_i^\alpha - t_j^\alpha\|^2\Big] + \\ & \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} \frac{1}{2}\Big[\rho\left(\kappa_{ij}^{\alpha\beta}\|R_i^\alpha \widetilde{R}_{ij}^{\alpha\beta} - R_j^\beta\|^2 + \tau_{ij}^{\alpha\beta}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha - t_j^\beta\|^2\right)\Big], \end{aligned} \tag{4.11}$$

where $\kappa_{ij}^{\alpha\alpha}$, $\tau_{ij}^{\alpha\alpha}$, $\kappa_{ij}^{\alpha\beta}$, $\tau_{ij}^{\alpha\beta}$ are weight factors and $\rho(\cdot):\mathbb{R}^+\to\mathbb{R}$ is the loss kernel.

For notational simplicity, $F(X)$ in Eq. (4.11) can be also rewritten as

$$F(X) = \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} F_{ij}^{\alpha\alpha}(X) + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} F_{ij}^{\alpha\beta}(X), \tag{4.12}$$

where

$$F_{ij}^{\alpha\alpha}(X) \triangleq \frac{1}{2}\kappa_{ij}^{\alpha\alpha}\|R_i^\alpha \widetilde{R}_{ij}^{\alpha\alpha} - R_j^\alpha\|^2 + \frac{1}{2}\tau_{ij}^{\alpha\alpha}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\alpha} + t_i^\alpha - t_j^\alpha\|^2, \tag{4.13}$$

and

$$F_{ij}^{\alpha\beta}(X) \triangleq \frac{1}{2}\rho\Big(\kappa_{ij}^{\alpha\beta}\|R_i^\alpha \widetilde{R}_{ij}^{\alpha\beta} - R_j^\beta\|^2 + \frac{1}{2}\tau_{ij}^{\alpha\beta}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha - t_j^\beta\|^2\Big), \tag{4.14}$$

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

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

4.5. The Majorization of Loss Kernels

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

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

$$\frac{1}{2}\|X\|^2_{M_{ij}^{\alpha\beta}} = \frac{1}{2}\kappa_{ij}^{\alpha\beta}\|R_i^\alpha \widetilde{R}_{ij}^{\alpha\beta} - R_j^\beta\|^2 + \frac{1}{2}\tau_{ij}^{\alpha\beta}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha - t_j^\beta\|^2. \tag{4.15}$$

Then, in terms of intra-node measurements with $\alpha = \beta$ and inter-node measurements with $\alpha \neq \beta$, $F_{ij}^{\alpha\alpha}(X)$ and $F_{ij}^{\alpha\beta}$ take the form of

$$F_{ij}^{\alpha\alpha}(X) \triangleq \frac{1}{2}\|X\|^2_{M_{ij}^{\alpha\alpha}} \tag{4.16}$$

and

$$F_{ij}^{\alpha\beta}(X) \triangleq \frac{1}{2}\rho\big(\|X\|^2_{M_{ij}^{\alpha\beta}}\big), \tag{4.17}$$

respectively. From Eqs. (4.13) and (4.14), we obtain an upper bound of $F_{ij}^{\alpha\alpha}(X)$ and $F_{ij}^{\alpha\beta}(X)$ as the following proposition states.

Proposition 4.5.1. Let $X^{(\mathsf{k})} = \begin{bmatrix} X^{1(\mathsf{k})} & \dots & X^{|\mathcal{A}|(\mathsf{k})} \end{bmatrix} \in \mathcal{X}$ with $X^{\alpha(\mathsf{k})} \in \mathcal{X}^\alpha$ be an iterate of Eq. (4.10). If $\rho(\cdot) : \mathbb{R}^+ \to \mathbb{R}$ is a loss kernel that satisfies Assumption 4.2, then

we obtain

$$\frac{1}{2}\omega_{ij}^{\alpha\beta(\mathsf{k})}\|X - X^{(\mathsf{k})}\|_{M_{ij}^{\alpha\beta}}^2 + \left\langle \nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), X - X^{(\mathsf{k})} \right\rangle + F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}) \geq F_{ij}^{\alpha\beta}(X) \tag{4.18}$$

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

$$\omega_{ij}^{\alpha\beta(\mathsf{k})} \triangleq \begin{cases} 1, & \alpha = \beta, \\ \nabla\rho\big(\|X^{(\mathsf{k})}\|_{M_{ij}^{\alpha\beta}}^2\big), & \alpha \neq \beta. \end{cases} \tag{4.19}$$

In Eq. (4.18), the equality "=" holds as long as $X = X^{(\mathsf{k})}$.

Proof. See Section 4.12.1 □

Note that $F(X)$, as is shown in Eq. (4.12), is equivalent to the sum of all $F_{ij}^{\alpha\alpha}(X)$ and $F_{ij}^{\alpha\beta}(X)$. Then, an immediate upper bound of $F(X)$ resulting from Proposition 4.5.1 is

$$\frac{1}{2}\big\|X - X^{(\mathsf{k})}\big\|_{M^{(\mathsf{k})}}^2 + \left\langle \nabla F(X^{(\mathsf{k})}), X - X^{(\mathsf{k})} \right\rangle + F(X^{(\mathsf{k})}) \geq F(X) \tag{4.20}$$

where $M^{(\mathsf{k})} \in \mathbb{R}^{(d+1)n\times(d+1)n}$ is a positive semidefinite matrix that is defined as

$$M^{(\mathsf{k})} \triangleq \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} M_{ij}^{\alpha\alpha} + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} \omega_{ij}^{\alpha\beta(\mathsf{k})} \cdot M_{ij}^{\alpha\beta} \in \mathbb{R}^{(d+1)n\times(d+1)n}. \tag{4.21}$$

In addition, the equality "=" in Eq. (4.20) holds as long as $X = X^{(\mathsf{k})}$.

Remark 4.2. If the loss kernel $\rho(\cdot)$ is non-trivial, $\omega_{ij}^{\alpha\beta(\mathsf{k})}$ is a function of $X^{(\mathsf{k})}$ as defined in Eq. (4.19), and $M^{(\mathsf{k})}$ is a positive semidefinite matrix depending on $X^{(\mathsf{k})}$ as well.

It is obvious that Eq. (4.20) has $X^\alpha \in \mathcal{X}^\alpha$ of different nodes coupled with each other, and as a result, is difficult to be used for distributed PGO. In spite of that, as is shown in the next sections, Eq. (4.20) is still useful for the development and analysis of our MM methods for distributed PGO.

4.6. The Majorization of Distributed Pose Graph Optimization

In this section, following a similar procedure to our previous works **[1, 59]**, we will present surrogate functions $G(X|X^{(\mathsf{k})})$ and $H(X|X^{(\mathsf{k})})$ that majorize the objective function $F(X)$. The surrogate functions $G(X|X^{(\mathsf{k})})$ and $H(X|X^{(\mathsf{k})})$ decouple unknown poses of different nodes, and thus, are critical to our MM methods for distributed PGO.

4.6.1. The Majorization of $F_{ij}^{\alpha\beta}(X)$

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

$$\frac{1}{2}\|B-C\|^2_{M_{ij}^{\alpha\beta}} \leq \|B-P\|^2_{M_{ij}^{\alpha\beta}} + \|C-P\|^2_{M_{ij}^{\alpha\beta}} \tag{4.22}$$

as long as $M_{ij}^{\alpha\beta} \in \mathbb{R}^{n\times n}$ is positive semidefinite, where "=" holds if

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

If we let $P = \mathbf{0}$, Eq. (4.22) becomes

$$\frac{1}{2}\|B-C\|^2_{M_{ij}^{\alpha\beta}} \leq \|B\|^2_{M_{ij}^{\alpha\beta}} + \|C\|^2_{M_{ij}^{\alpha\beta}}. \tag{4.23}$$

Applying Eq. (4.23) on the right-hand side of Eq. (4.15), we obtain

$$\begin{aligned}\frac{1}{2}\|X\|^2_{M_{ij}^{\alpha\beta}} \leq& \kappa_{ij}^{\alpha\beta}\|R_i^\alpha \widetilde{R}_{ij}^{\alpha\beta}\|^2 + \kappa_{ij}^{\alpha\beta}\|R_j^\beta\|^2 + \tau_{ij}^{\alpha\beta}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha\|^2 + \tau_{ij}^{\alpha\beta}\|t_j^\beta\|^2 \\ =& \kappa_{ij}^{\alpha\beta}\|R_i^\alpha\|^2 + \kappa_{ij}^{\alpha\beta}\|R_j^\beta\|^2 + \tau_{ij}^{\alpha\beta}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha\|^2 + \tau_{ij}^{\alpha\beta}\|t_j^\beta\|^2,\end{aligned} \tag{4.24}$$

where the last equality is due to $\big(\widetilde{R}_{ij}^{\alpha\beta}\big)^\top \widetilde{R}_{ij}^{\alpha\beta} = \widetilde{R}_{ij}^{\alpha\beta}\big(\widetilde{R}_{ij}^{\alpha\beta}\big)^\top = \mathbf{I}$. Furthermore, there exists a positive semidefinite matrix $\Omega_{ij}^{\alpha\beta} \in \mathbb{R}^{(d+1)n\times(d+1)n}$ such that the right-hand side of Eq. (4.24) can be rewritten as

$$\frac{1}{2}\|X\|^2_{\Omega_{ij}^{\alpha\beta}} = \kappa_{ij}^{\alpha\beta}\|R_i^\alpha\|^2 + \kappa_{ij}^{\alpha\beta}\|R_j^\beta\|^2 + \tau_{ij}^{\alpha\beta}\|R_i^\alpha \tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha\|^2 + \tau_{ij}^{\alpha\beta}\|t_j^\beta\|^2, \tag{4.25}$$

where $\Omega_{ij}^{\alpha\beta}$ is a block diagonal matrix decoupling unknown poses of different nodes. Replacing the right-hand side of Eq. (4.24) with Eq. (4.25) results in

$$\frac{1}{2}\|X\|^2_{M_{ij}^{\alpha\beta}} \leq \frac{1}{2}\|X\|^2_{\Omega_{ij}^{\alpha\beta}}$$

for any $X \in \mathbb{R}^{d\times(d+1)n}$, which suggests

$$\Omega_{ij}^{\alpha\beta} \succeq M_{ij}^{\alpha\beta}. \tag{4.26}$$

With $\Omega_{ij}^{\alpha\beta} \in \mathbb{R}^{(d+1)n\times(d+1)n}$ in Eqs. (4.25) and (4.26), we define $E_{ij}^{\alpha\beta}(\cdot|X^{(\mathsf{k})}) : \mathbb{R}^{d\times(d+1)n} \to \mathbb{R}$:

$$E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})}) \triangleq \frac{1}{2}\omega_{ij}^{\alpha\beta(\mathsf{k})}\|X - X^{(\mathsf{k})}\|^2_{\Omega_{ij}^{\alpha\beta}} + \big\langle \nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), X - X^{(\mathsf{k})}\big\rangle + F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), \tag{4.27}$$

where $\omega_{ij}^{\alpha\beta(\mathsf{k})}$ is given in Eq. (4.19). From the equation above, it can be concluded that $E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})})$ majorizes $F_{ij}^{\alpha\beta}(X)$ as the following proposition states, which is important for the construction of surrogate functions for distributed PGO.

Proposition 4.6.1. Given any nodes $\alpha, \beta \in \mathcal{A}$ with either $\alpha = \beta$ or $\alpha \neq \beta$, if $\rho(\cdot) : \mathbb{R}^+ \to \mathbb{R}$ is a loss kernel that satisfies Assumption 4.2, then we obtain

$$E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})}) \geq F_{ij}^{\alpha\beta}(X). \tag{4.28}$$

for any $X \in \mathbb{R}^{d\times(d+1)n}$. In the equation above, the equality "=" holds if $X = X^{(\mathsf{k})}$.

Proof. See Section 4.12.2 □

4.6.2. The Majorization of $F(X)$

From Proposition 4.6.1, it is immediate to construct surrogate functions that majorize $F(X)$ in Eqs. (4.11) and (4.12) as the following proposition states.

Proposition 4.6.2. Let $X^{(\mathsf{k})} = \begin{bmatrix} X^{1(\mathsf{k})} & \dots & X^{|\mathcal{A}|(\mathsf{k})} \end{bmatrix} \in \mathcal{X}$ with $X^{\alpha(\mathsf{k})} \in \mathcal{X}^\alpha$ be an iterate of $X \in \mathcal{X}$ for Eq. (4.10). Let $G(\cdot|X^{(\mathsf{k})}) : \mathbb{R}^{d\times(d+1)n} \to \mathbb{R}$ be a function that is defined as

$$G(X|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} F_{ij}^{\alpha\alpha}(X) + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})}) + \frac{\xi}{2}\left\|X - X^{(\mathsf{k})}\right\|^2 \tag{4.29}$$

where $\xi \in \mathbb{R}$ and $\xi \geq 0$. Then, we have the following results:

(a) For any node $\alpha \in \mathcal{A}$, there exists positive-semidefinite matrices $\Gamma^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)n_\alpha \times (d+1)n_\alpha}$ such that $G(X|X^{(\mathsf{k})})$ is equivalent to

$$G(X|X^{(\mathsf{k})}) = \sum_{\alpha \in \mathcal{A}} G^\alpha(X^\alpha|X^{(\mathsf{k})}) + F(X^{(\mathsf{k})}), \tag{4.30}$$

where $G^\alpha(X^\alpha|X^{(\mathsf{k})})$ is defined as

$$G^\alpha(X^\alpha|X^{(\mathsf{k})}) = \frac{1}{2}\|X^\alpha - X^{\alpha(\mathsf{k})}\|^2_{\Gamma^{\alpha(\mathsf{k})}} + \left\langle \nabla_{X^\alpha} F(X^{(\mathsf{k})}),\, X^\alpha - X^{\alpha(\mathsf{k})} \right\rangle. \tag{4.31}$$

In Eq. (4.31), $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$ is the Euclidean gradient of $F(X)$ with respect to $X^\alpha \in \mathcal{X}^\alpha$ at $X^{(\mathsf{k})} \in \mathcal{X}$.

(b) $G(X|X^{(\mathsf{k})})$ is a proximal operator of $F(X)$ at $X^{(\mathsf{k})} \in \mathcal{X}$ and can be written as

$$G(X|X^{(\mathsf{k})}) = \frac{1}{2}\|X - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}} + \left\langle \nabla F(X^{(\mathsf{k})}), X - X^{(\mathsf{k})} \right\rangle + F(X^{(\mathsf{k})}), \tag{4.32}$$

where $\Gamma^{(\mathsf{k})} \in \mathbb{R}^{(d+1)n \times (d+1)n}$ is a block diagonal matrix

$$\Gamma^{(\mathsf{k})} \triangleq \operatorname{diag}\left\{\Gamma^{1(\mathsf{k})}, \cdots, \Gamma^{|\mathcal{A}|(\mathsf{k})}\right\} \in \mathbb{R}^{(d+1)n \times (d+1)n}, \tag{4.33}$$

and $\nabla F(X^{(\mathsf{k})}) \in \mathbb{R}^{d \times (d+1)n}$ is the Euclidean gradient of $F(X)$ at $X^{(\mathsf{k})} \in \mathcal{X}$. Furthermore, we have

$$G(X|X^{(\mathsf{k})}) \geq F(X) \tag{4.34}$$

where the equality "=" holds if $X = X^{(\mathsf{k})}$.

(c) $\Gamma^{(\mathsf{k})} \succeq M^{(\mathsf{k})}$ where $M^{(\mathsf{k})}$ is given in Eq. (4.21).

(d) $\Gamma^{(\mathsf{k})}$ is bounded, i.e., there exists a constant positive-semidefinite matrix $\Gamma \in \mathbb{R}^{(d+1)n\times(d+1)n}$ such that $\Gamma \succeq \Gamma^{(\mathsf{k})}$ holds for any $\mathsf{k} \geq 0$.

Proof. See Section 4.12.3 □

From Proposition 4.6.2, it is known that $G(X|X^{(\mathsf{k})})$ in Eqs. (4.29) and (4.30) is a proximal operator as well as an upper bound of $F(X)$. Instead of depending on $X = \begin{bmatrix} X^1 & \dots & X^{|\mathcal{A}|} \end{bmatrix} \in \mathcal{X}$ of all the nodes, each $G^\alpha(X^\alpha|X^{(\mathsf{k})})$ in Eq. (4.30) is a function of $X^\alpha \in \mathcal{X}^\alpha \subset \mathbb{R}^{d\times(d+1)n_\alpha}$ within a single node $\alpha \in \mathcal{A}$, which makes $G(X|X^{(\mathsf{k})})$ well-suited for distributed PGO.

If substituting Eqs. (4.27) and (4.28) into Eq. (4.29) to replace $F_{ij}^{\alpha\alpha}(X|X^{(\mathsf{k})})$ with $E_{ij}^{\alpha\alpha}(X|X^{(\mathsf{k})})$, we have $F(X)$ as well as $G(X|X^{(\mathsf{k})})$ further majorized as the following proposition states.

Proposition 4.6.3. Let $X^{(\mathsf{k})} = \begin{bmatrix} X^{1(\mathsf{k})} & \dots & X^{|\mathcal{A}|(\mathsf{k})} \end{bmatrix} \in \mathcal{X}$ with $X^{\alpha(\mathsf{k})} \in \mathcal{X}^\alpha$ be an iterate of $X \in \mathcal{X}$ for Eq. (4.10), and $X_i^{\alpha(\mathsf{k})} = \begin{bmatrix} t^{\alpha(\mathsf{k})} & R^{\alpha(\mathsf{k})} \end{bmatrix} \in SE(d)$ be the corresponding iterate of $X_i^\alpha \in SE(d)$. Let $H(\cdot|X^{(\mathsf{k})}) : \mathbb{R}^{d\times(d+1)n} \to \mathbb{R}$ be a function that is defined as

$$H(X|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} E_{ij}^{\alpha\alpha}(X|X^{(\mathsf{k})}) + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})}) + \frac{\zeta}{2}\left\|X - X^{(\mathsf{k})}\right\|^2. \tag{4.35}$$

In Eq. (4.35), $\zeta \in \mathbb{R}$ and $\zeta \geq \xi \geq 0$ where $\xi \in \mathbb{R}$ is given in $G(X|X^{(\mathsf{k})})$ of Eq. (4.29). Then, we have the following results:

(a) For any node $\alpha \in \mathcal{A}$ and $i \in \{1, \cdots, n_\alpha\}$, there exists positive-semidefinite matrices $\Pi^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)n_\alpha \times (d+1)n_\alpha}$ and $\Pi_i^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)\times(d+1)}$ such that

$$H(X|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} H^\alpha(X^\alpha|X^{(\mathsf{k})}) + F(X^{(\mathsf{k})}) \tag{4.36}$$

and

$$H^\alpha(X^\alpha|X^{(\mathsf{k})}) = \sum_{i=1}^{n_\alpha} H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})}), \tag{4.37}$$

where $H^\alpha(X^\alpha|X^{(\mathsf{k})})$ and $H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})})$ are defined as

$$H^\alpha(X^\alpha|X^{(\mathsf{k})}) = \frac{1}{2}\|X^\alpha - X^{\alpha(\mathsf{k})}\|^2_{\Pi^{\alpha(\mathsf{k})}} + \left\langle \nabla_{X^\alpha} F(X^{(\mathsf{k})}),\, X^\alpha - X^{\alpha(\mathsf{k})}\right\rangle \tag{4.38}$$

and

$$H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})}) = \frac{1}{2}\|X_i^\alpha - X_i^{\alpha(\mathsf{k})}\|^2_{\Pi_i^{\alpha(\mathsf{k})}} + \left\langle \nabla_{X_i^\alpha} F(X^{(\mathsf{k})}),\, X_i^\alpha - X_i^{\alpha(\mathsf{k})}\right\rangle, \tag{4.39}$$

respectively. In Eqs. (4.38) and (4.39), $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$ and $\nabla_{X_i^\alpha} F(X^{(\mathsf{k})})$ are the Euclidean gradients of $F(X)$ with respect to $X^\alpha \in \mathcal{X}^\alpha$ and $X_i^\alpha \in SE(d)$ at $X^{(\mathsf{k})} \in \mathcal{X}$, respectively.

(b) $H(X|X^{(\mathsf{k})})$ is a proximal operator of $F(X)$ at $X^{(\mathsf{k})} \in \mathcal{X}$ and can be written as

$$H(X|X^{(\mathsf{k})}) = \frac{1}{2}\|X - X^{(\mathsf{k})}\|^2_{\Pi^{(\mathsf{k})}} + \left\langle \nabla F(X^{(\mathsf{k})}), X - X^{(\mathsf{k})}\right\rangle + F(X^{(\mathsf{k})}), \tag{4.40}$$

where $\Pi^{(\mathsf{k})} \in \mathbb{R}^{(d+1)n\times(d+1)n}$ is a block diagonal matrix

$$\Pi^{(\mathsf{k})} \triangleq \operatorname{diag}\left\{\Pi^{1(\mathsf{k})}, \cdots, \Pi^{|\mathcal{A}|(\mathsf{k})}\right\} \in \mathbb{R}^{(d+1)n\times(d+1)n}, \tag{4.41}$$

and $\nabla F(X^{(\mathsf{k})}) \in \mathbb{R}^{d\times(d+1)n}$ is the Euclidean gradient of $F(X)$ at $X^{(\mathsf{k})} \in \mathcal{X}$. Furthermore, we have

$$H(X|X^{(\mathsf{k})}) \geq G(X|X^{\mathsf{k}}) \geq F(X) \tag{4.42}$$

where the equality "=" holds if $X = X^{(\mathsf{k})}$.

(c) $\Pi^{(\mathsf{k})} \succeq \Gamma^{(\mathsf{k})} \succeq M^{(\mathsf{k})}$ where $M^{(\mathsf{k})}$ and $\Gamma^{(\mathsf{k})}$ are given in Eqs. (4.21) and (4.33), respectively.

(d) $\Pi^{(\mathsf{k})}$ is bounded, i.e., there exists a constant positive-semidefinite matrix $\Pi \in \mathbb{R}^{(d+1)n\times(d+1)n}$ such that $\Pi \succeq \Pi^{(\mathsf{k})}$ holds for any $\mathsf{k} \geq 0$.

(e) $H^\alpha(X^\alpha|X^{(\mathsf{k})}) \geq G^\alpha(X^\alpha|X^{(\mathsf{k})})$ where $G^\alpha(X^\alpha|X^{(\mathsf{k})})$ is given in Eq. (4.31) and the equality "=" holds as long as $X^\alpha = X^{\alpha(\mathsf{k})}$.

Proof. The proof is similar to that of Proposition 4.6.2. □

Remark 4.3. As a result of Eqs. (4.36) and (4.37), $H(X|X^{(\mathsf{k})})$ can be rewritten as the sum of $H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})})$:

$$H(X|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}}\sum_{i=1}^{n_\alpha} H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})}) + F(X^{(\mathsf{k})}),$$

where note that each $H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})})$ relies on a single pose $X_i^\alpha \in SE(d) \subset \mathbb{R}^{d\times(d+1)}$. In Sections 4.7 to 4.9, we will exploit this decomposition of $H(X|X^{(\mathsf{k})})$ to improve the computational efficiency of distributed PGO.

From Eqs. (4.32) and (4.40), the Euclidean gradient $\nabla F(X)$ of $F(X)$ is needed in $G(X|X^{(\mathsf{k})})$ and $H(X|X^{(\mathsf{k})})$. From Eqs. (4.12) to (4.14), it can be shown that $\nabla_{X^\alpha} F(X)$,

i.e., the Euclidean gradient of $F(X)$ with respect to $X^\alpha \in \mathcal{X}^\alpha$, is only related with $F_{ij}^{\alpha\alpha}(X)$, $F_{ij}^{\alpha\beta}(X)$ and $F_{ji}^{\beta\alpha}(X)$, which suggests that

$$\nabla_{X^\alpha} F(X^{(\mathsf{k})}) = \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} \nabla_{X^\alpha} F_{ij}^{\alpha\alpha}(X^{(\mathsf{k})}) + \sum_{\beta\in\mathcal{N}_-^\alpha} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} \nabla_{X^\alpha} F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}) + \sum_{\beta\in\mathcal{N}_+^\alpha} \sum_{(j,i)\in\overrightarrow{\mathcal{E}}^{\beta\alpha}} \nabla_{X^\alpha} F_{ji}^{\beta\alpha}(X^{(\mathsf{k})}). \tag{4.43}$$

In addition, Eqs. (4.13) and (4.14) indicate that $F_{ij}^{\alpha\alpha}(X)$ depends X_i^α and X_j^α, and $F_{ij}^{\alpha\beta}(X)$ and $F_{ji}^{\beta\alpha}$ depend on X_i^α and X_j^β. Therefore, $\nabla_{X^\alpha} F(X)$ can be computed using Eq. (4.43) in a distributed setting as long as each node $\alpha \in \mathcal{A}$ can communicate with its neighbors $\beta \in \mathcal{N}^\alpha$. If $\nabla_{X^\alpha} F(X)$ is known for each node $\alpha \in \mathcal{A}$, then

$$\nabla F(X) \triangleq \begin{bmatrix} \nabla_{X^1} F(X) & \cdots & \nabla_{X^{|\mathcal{A}|}} F(X) \end{bmatrix}$$

and

$$\nabla_{X_i^\alpha} F(X) \triangleq \begin{bmatrix} \nabla_{t_i^\alpha} F(X) & \nabla_{R_i^\alpha} F(X) \end{bmatrix}$$

are immediately known. Thus, it can be concluded from Eqs. (4.31), (4.38) and (4.39) that $G^\alpha(X^\alpha|X^{(\mathsf{k})})$ in $G(X|X^{(\mathsf{k})})$ as well as $H^\alpha(X^\alpha|X^{(\mathsf{k})})$ and $H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})})$ in $H(X|X^{(\mathsf{k})})$ can be constructed in a distributed setting with one communication round between neighboring nodes α and β.

4.6.3. The Formulation of $\Gamma^{\alpha(\mathsf{k})}$ in Eq. (4.31)

From Eqs. (4.15), (4.25), (4.29) and (4.31), a straightforward but tedious mathematical manipulation results in

$$
\begin{aligned}
&\frac{1}{2}\|X^{\alpha}-X^{\alpha(\mathsf{k})}\|^2_{\Gamma^{\alpha(\mathsf{k})}}\\
=&\sum_{i=1}^{n_\alpha}\frac{\xi}{2}\Big(\big\|R_i^{\alpha}-R_i^{\alpha(\mathsf{k})}\big\|^2+\big\|t_i^{\alpha}-t_i^{\alpha(\mathsf{k})}\big\|^2\Big)\\
&\sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}}\frac{1}{2}\Big(\kappa_{ij}^{\alpha\alpha}\big\|(R_i^{\alpha}-R_i^{\alpha(\mathsf{k})})\widetilde{R}_{ij}^{\alpha\alpha}-(R_j^{\alpha}-R_j^{\alpha(\mathsf{k})})\big\|^2\\
&\qquad\qquad\tau_{ij}^{\alpha\alpha}\big\|(R_i^{\alpha}-R_i^{\alpha(\mathsf{k})})\tilde{t}_{ij}^{\alpha\alpha}+t_i^{\alpha}-t_i^{\alpha(\mathsf{k})}-(t_j^{\alpha}-t_j^{\alpha(\mathsf{k})})\big\|^2\Big)+\\
&\sum_{\beta\in\mathcal{N}_-^{\alpha}}\sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}}\omega_{ij}^{\alpha\beta(\mathsf{k})}\Big(\kappa_{ij}^{\alpha\beta}\big\|R_i^{\alpha}-R_i^{\alpha(\mathsf{k})}\big\|^2+\\
&\qquad\qquad\tau_{ij}^{\alpha\beta}\big\|\big(R_i^{\alpha}-R_i^{\alpha(\mathsf{k})}\big)\tilde{t}_{ij}^{\alpha\beta}+t_i^{\alpha}-t_i^{\alpha(\mathsf{k})}\big\|^2\Big)+\\
&\sum_{\beta\in\mathcal{N}_+^{\alpha}}\sum_{(j,i)\in\overrightarrow{\mathcal{E}}^{\beta\alpha}}\omega_{ji}^{\beta\alpha(\mathsf{k})}\Big(\kappa_{ji}^{\beta\alpha}\big\|R_i^{\alpha}-R_i^{\alpha(\mathsf{k})}\big\|^2+\tau_{ji}^{\beta\alpha}\big\|t_i^{\alpha}-t_i^{\alpha(\mathsf{k})}\big\|^2\Big).
\end{aligned}
\tag{4.44}
$$

For notational clarity, we introduce

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

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

$$\mathcal{N}_{i-}^{\alpha}\triangleq\{\beta\in\mathcal{A}|\exists(i,\,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}\text{ and }\beta\neq\alpha\},$$

$$\mathcal{N}_{i+}^{\alpha}\triangleq\{\beta\in\mathcal{A}|\exists(j,\,i)\in\overrightarrow{\mathcal{E}}^{\beta\alpha}\text{ and }\beta\neq\alpha\},$$

$$\mathcal{E}_i^{\alpha\beta}\triangleq\overrightarrow{\mathcal{E}}_{i-}^{\alpha\beta}\cup\overrightarrow{\mathcal{E}}_{i+}^{\alpha\beta},$$

$$\mathcal{N}_i^{\alpha}\triangleq\mathcal{N}_{i-}^{\alpha}\cup\mathcal{N}_{i+}^{\alpha}.$$

In addition, we define $\kappa_{ji}^{\beta\alpha} \triangleq \kappa_{ij}^{\alpha\beta}$, $\tau_{ji}^{\beta\alpha} \triangleq \tau_{ij}^{\alpha\beta}$, $\omega_{ij}^{\alpha\beta(\mathsf{k})} \triangleq \omega_{ji}^{\beta\alpha(\mathsf{k})}$, and

$$\kappa_{ij}^{\alpha\beta(k)} \triangleq \omega_{ij}^{\alpha\beta(\mathsf{k})} \cdot \kappa_{ij}^{\alpha\beta}, \tag{4.45}$$

$$\tau_{ij}^{\alpha\beta(k)} \triangleq \omega_{ij}^{\alpha\beta(\mathsf{k})} \cdot \tau_{ij}^{\alpha\beta} \tag{4.46}$$

for any $(i,\, j) \in \overrightarrow{\mathcal{E}}^{\alpha\beta}$. Then, Eq. (4.44) indicates that $\Gamma^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)n_\alpha \times (d+1)n_\alpha}$ in Eq. (4.31) takes the form as

$$\Gamma^{\alpha(\mathsf{k})} = \begin{bmatrix} \Gamma^{\tau,\alpha(\mathsf{k})} & \Gamma^{\nu,\alpha(\mathsf{k})} \\ \Gamma^{\nu,\alpha(\mathsf{k})\top} & \Gamma^{\kappa,\alpha(\mathsf{k})} \end{bmatrix}, \tag{4.47}$$

where $\Gamma^{\tau,\alpha(\mathsf{k})} \in \mathbb{R}^{n_\alpha \times n_\alpha}$, $\Gamma^{\nu,\alpha(\mathsf{k})} \in \mathbb{R}^{n_\alpha \times dn_\alpha}$ and $\Gamma^{\kappa,\alpha(\mathsf{k})} \in \mathbb{R}^{dn_\alpha \times dn_\alpha}$ are sparse matrices that are defined as

$$\left[\Gamma^{\tau,\alpha(\mathsf{k})}\right]_{ij} = \begin{cases} \xi + \sum\limits_{e \in \mathcal{E}_i^{\alpha\alpha}} \tau_e^{\alpha\alpha} + \sum\limits_{\beta \in \mathcal{N}_i^\alpha} \sum\limits_{e \in \mathcal{E}_i^{\alpha\beta}} 2\tau_e^{\alpha\beta(\mathsf{k})}, & i = j, \\ -\tau_{ij}^{\alpha\alpha}, & (i,j) \in \overrightarrow{\mathcal{E}}^{\alpha\alpha}, \\ -\tau_{ji}^{\alpha\alpha}, & (j,i) \in \overrightarrow{\mathcal{E}}^{\alpha\alpha}, \\ 0, & \text{otherwise}, \end{cases}$$

$$\left[\Gamma^{\nu,\alpha(\mathsf{k})}\right]_{ij} = \begin{cases} \sum\limits_{e \in \mathcal{E}_{i-}^{\alpha\alpha}} \tau_e^{\alpha\alpha} \tilde{t}_e^{\alpha\alpha\top} + \sum\limits_{\beta \in \mathcal{N}_{i-}^\alpha} \sum\limits_{e \in \mathcal{E}_{i-}^{\alpha\beta}} 2\tau_e^{\alpha\beta(\mathsf{k})} \tilde{t}_e^{\alpha\beta\top}, & i = j, \\ -\tau_{ji}^{\alpha\alpha} \tilde{t}_{ji}^{\alpha\alpha\top}, & (j,i) \in \overrightarrow{\mathcal{E}}^{\alpha\alpha}, \\ 0, & \text{otherwise}, \end{cases}$$

$$\Big[\Gamma^{\kappa,\alpha(\mathsf{k})}\Big]_{ij}=\begin{cases}\xi\cdot\mathbf{I}+\sum\limits_{e\in\mathcal{E}_i^{\alpha\alpha}}\kappa_e^{\alpha\alpha}\cdot\mathbf{I}+\sum\limits_{e\in\mathcal{E}_{i-}^{\alpha\alpha}}\tau_e^{\alpha\alpha}\cdot\tilde{t}_e^{\alpha\alpha}\tilde{t}_e^{\alpha\alpha\top}+ & \\ \qquad\sum\limits_{\beta\in\mathcal{N}_i^\alpha}2\Big(\sum\limits_{e\in\mathcal{E}_i^{\alpha\beta}}\kappa_e^{\alpha\beta(\mathsf{k})}\cdot\mathbf{I}+\sum\limits_{e\in\mathcal{E}_{i-}^{\alpha\beta}}\tau_e^{\alpha\beta(\mathsf{k})}\cdot\tilde{t}_e^{\alpha\beta}\tilde{t}_e^{\alpha\beta\top}\Big), & i=j,\\ -\kappa_{ij}^{\alpha\alpha}\cdot\widetilde{R}_{ij}^{\alpha\alpha}, & (i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha},\\ -\kappa_{ji}^{\alpha\alpha}\cdot\widetilde{R}_{ij}^{\alpha\alpha\top}, & (j,i)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha},\\ 0, & \text{otherwise.}\end{cases}$$

4.6.4. The Formulation of $\Pi^{\alpha(\mathsf{k})}$ and $\Pi_i^{\alpha(\mathsf{k})}$ in Eqs. (4.38) and (4.39)

Similar to Eq. (4.44), it can be shown from Eqs. (4.15), (4.25), (4.35), (4.37) and (4.38) that

$$\begin{aligned}&\frac{1}{2}\|X_i^\alpha-X_i^{\alpha(\mathsf{k})}\|^2_{\Pi_i^{\alpha}(\mathsf{k})}\\ =&\frac{\zeta}{2}\Big(\big\|R_i^\alpha-R_i^{\alpha(\mathsf{k})}\big\|^2+\big\|t_i^\alpha-t_i^{\alpha(\mathsf{k})}\big\|^2\Big)+\\ &\sum_{(i,j)\in\overrightarrow{\mathcal{E}}_{i-}^{\alpha\alpha}}\Big(\kappa_{ij}^{\alpha\alpha}\big\|R_i^\alpha-R_i^{\alpha(\mathsf{k})}\big\|^2+\tau_{ij}^{\alpha\alpha}\big\|\big(R_i^\alpha-R_i^{\alpha(\mathsf{k})}\big)\tilde{t}_{ij}^{\alpha\alpha}+t_i^\alpha-t_i^{\alpha(\mathsf{k})}\big\|^2\Big)+\\ &\sum_{(i,j)\in\overrightarrow{\mathcal{E}}_{i+}^{\alpha\alpha}}\Big(\kappa_{ji}^{\alpha\alpha}\big\|R_i^\alpha-R_i^{\alpha(\mathsf{k})}\big\|^2+\tau_{ji}^{\alpha\alpha}\big\|t_i^\alpha-t_i^{\alpha(\mathsf{k})}\big\|^2\Big)+\\ &\sum_{\beta\in\mathcal{N}_{i-}^\alpha}\sum_{(i,j)\in\overrightarrow{\mathcal{E}}_{i-}^{\alpha\beta}}\omega_{ij}^{\alpha\beta(\mathsf{k})}\Big(\kappa_{ij}^{\alpha\beta}\big\|R_i^\alpha-R_i^{\alpha(\mathsf{k})}\big\|^2+\\ &\qquad\qquad\tau_{ij}^{\alpha\beta}\big\|\big(R_i^\alpha-R_i^{\alpha(\mathsf{k})}\big)\tilde{t}_{ij}^{\alpha\beta}+t_i^\alpha-t_i^{\alpha(\mathsf{k})}\big\|^2\Big)+\\ &\sum_{\beta\in\mathcal{N}_{i+}^\alpha}\sum_{(j,i)\in\overrightarrow{\mathcal{E}}_{i+}^{\beta\alpha}}\omega_{ji}^{\beta\alpha(\mathsf{k})}\Big(\kappa_{ji}^{\beta\alpha}\big\|R_i^\alpha-R_i^{\alpha(\mathsf{k})}\big\|^2+\tau_{ji}^{\beta\alpha}\big\|t_i^\alpha-t_i^{\alpha(\mathsf{k})}\big\|^2\Big).\end{aligned}\tag{4.48}$$

From Eq. (4.48), $\Pi_i^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)\times(d+1)}$ in Eq. (4.39) can be written as

$$\Pi_i^{\alpha(\mathsf{k})} = \begin{bmatrix} \Pi_i^{\tau,\alpha(\mathsf{k})} & \Pi_i^{\nu,\alpha(\mathsf{k})} \\ \Pi_i^{\nu,\alpha(\mathsf{k})^\top} & \Pi_i^{\kappa,\alpha(\mathsf{k})} \end{bmatrix}, \tag{4.49}$$

where $\Pi_i^{\tau,\alpha(\mathsf{k})} \in \mathbb{R}$, $\Pi_i^{\nu,\alpha(\mathsf{k})} \in \mathbb{R}^{1\times d}$ and $\Pi_i^{\kappa,\alpha(\mathsf{k})} \in \mathbb{R}^{d\times d}$ are defined as

$$\Pi_i^{\tau,\alpha(\mathsf{k})} = \zeta + \sum_{e\in\mathcal{E}_i^{\alpha\alpha}} 2\tau_e^{\alpha\alpha} + \sum_{\beta\in\mathcal{N}_i^\alpha} \sum_{e\in\mathcal{E}_i^{\alpha\beta}} 2\tau_e^{\alpha\beta(\mathsf{k})}, \tag{4.50}$$

$$\Pi_i^{\nu,\alpha(\mathsf{k})} = \sum_{e\in\mathcal{E}_{i-}^{\alpha\alpha}} 2\tau_e^{\alpha\alpha}\tilde{t}_e^{\alpha\alpha\top} + \sum_{\beta\in\mathcal{N}_{i-}^\alpha} \sum_{e\in\mathcal{E}_{i-}^{\alpha\beta}} 2\tau_e^{\alpha\beta(\mathsf{k})}\tilde{t}_e^{\alpha\beta\top}, \tag{4.51}$$

$$\begin{aligned} \Pi_i^{\kappa,\alpha(\mathsf{k})} = \zeta\cdot\mathbf{I} + \sum_{e\in\mathcal{E}_i^{\alpha\alpha}} 2\kappa_e^{\alpha\alpha}\cdot\mathbf{I} + \sum_{e\in\mathcal{E}_{i-}^{\alpha\alpha}} 2\tau_e^{\alpha\alpha}\cdot\tilde{t}_e^{\alpha\alpha}\tilde{t}_e^{\alpha\alpha\top} + \\ \sum_{\beta\in\mathcal{N}_i^\alpha}\Big(\sum_{e\in\mathcal{E}_i^{\alpha\beta}} 2\kappa_e^{\alpha\beta(\mathsf{k})}\cdot\mathbf{I} + \sum_{e\in\mathcal{E}_{i-}^{\alpha\beta}} 2\tau_e^{\alpha\beta(\mathsf{k})}\cdot\tilde{t}_e^{\alpha\beta}\tilde{t}_e^{\alpha\beta\top}\Big), \end{aligned} \tag{4.52}$$

where $\kappa_{ij}^{\alpha\beta(\mathsf{k})}$ and $\tau_{ij}^{\alpha\beta(\mathsf{k})}$ are given in Eqs. (4.45) and (4.46), respectively.

Similar to $\Gamma^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)n_\alpha\times(d+1)n_\alpha}$ in Eq. (4.47), $\Pi^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)n_\alpha\times(d+1)n_\alpha}$ in Eq. (4.38) also takes the form as

$$\Pi^{\alpha(\mathsf{k})} = \begin{bmatrix} \Pi^{\tau,\alpha(\mathsf{k})} & \Pi^{\nu,\alpha(\mathsf{k})} \\ \Pi^{\nu,\alpha(\mathsf{k})^\top} & \Pi^{\kappa,\alpha(\mathsf{k})} \end{bmatrix}, \tag{4.53}$$

where $\Pi^{\tau,\alpha(\mathsf{k})} \in \mathbb{R}^{n_\alpha \times n_\alpha}$, $\Pi^{\nu,\alpha(\mathsf{k})} \in \mathbb{R}^{n_\alpha \times dn_\alpha}$ and $\Pi^{\kappa,\alpha(\mathsf{k})} \in \mathbb{R}^{dn_\alpha \times dn_\alpha}$ are sparse matrices. Following Eqs. (4.37) to (4.39), it is straightforward to show that

$$\frac{1}{2}\|X^\alpha - X^{\alpha(\mathsf{k})}\|^2_{\Pi^{\alpha(\mathsf{k})}} = \sum_{i=1}^{n_\alpha} \frac{1}{2}\|X_i^\alpha - X_i^{\alpha(\mathsf{k})}\|^2_{\Pi_i^{\alpha(\mathsf{k})}}.$$

From the equation above, $\Pi^{\tau,\alpha(\mathsf{k})} \in \mathbb{R}^{n_\alpha \times n_\alpha}$, $\Pi^{\nu,\alpha(\mathsf{k})} \in \mathbb{R}^{n_\alpha \times dn_\alpha}$ and $\Pi^{\kappa,\alpha(\mathsf{k})} \in \mathbb{R}^{dn_\alpha \times dn_\alpha}$ in Eq. (4.53) are defined as

$$\left[\Pi^{\tau,\alpha(\mathsf{k})}\right]_{ij} = \begin{cases} \Pi_i^{\tau,\alpha(\mathsf{k})}, & i = j, \\ 0, & \text{otherwise}, \end{cases}$$

$$\left[\Gamma^{\nu,\alpha(\mathsf{k})}\right]_{ij} = \begin{cases} \Pi_i^{\nu,\alpha(\mathsf{k})}, & i = j, \\ 0, & \text{otherwise}, \end{cases}$$

$$\left[\Gamma^{\kappa,\alpha(\mathsf{k})}\right]_{ij} = \begin{cases} \Pi_i^{\kappa,\alpha(\mathsf{k})}, & i = j, \\ 0, & \text{otherwise}, \end{cases}$$

where $\Pi_i^{\tau,\alpha(\mathsf{k})} \in \mathbb{R}$, $\Pi_i^{\nu,\alpha(\mathsf{k})} \in \mathbb{R}^{1\times d}$ and $\Pi_i^{\kappa,\alpha(\mathsf{k})} \in \mathbb{R}^{d\times d}$ are given in Eqs. (4.50) to (4.52).

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

4.7. The Majorization Minimization Method for Distributed Pose Graph Optimization

In distributed optimization, MM methods are one of the most popular first-order optimization methods [**117**, **118**]. As mentioned before, MM methods solve an optimization

problem by iteratively minimizing an upper bound of the objective function such that the objective value is either decreased or unchanged.

4.7.1. Update Rule

In Section 4.6, it has been proved that $G(X|X^{(\mathsf{k})})$ and $H(X|X^{(\mathsf{k})})$ are proximal operators of $F(X)$ such that

$$H(X|X^{(\mathsf{k})}) \geq G(X|X^{(\mathsf{k})}) \geq F(X),$$

and

$$H(X^{(\mathsf{k})}|X^{(\mathsf{k})}) = G(X^{(\mathsf{k})}|X^{(\mathsf{k})}) = F(X^{(\mathsf{k})}).$$

Following the notion of MM methods **[117]**, we implement an update rule as the following

$$X^{(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X\in\mathcal{X}} H(X|X^{(\mathsf{k})}), \tag{4.54}$$

and

$$X^{(\mathsf{k}+1)} \leftarrow \arg\min_{X\in\mathcal{X}} G(X|X^{(\mathsf{k})}) \tag{4.55}$$

which results in

$$F(X^{(\mathsf{k})}) = H(X^{(\mathsf{k})}|X^{(\mathsf{k})}) \geq H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \geq F(X^{(\mathsf{k}+\frac{1}{2})}), \tag{4.56}$$

and

$$F(X^{(\mathsf{k})}) = G(X^{(\mathsf{k})}|X^{(\mathsf{k})}) \geq G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \geq F(X^{(\mathsf{k}+1)}) \tag{4.57}$$

respectively. From Eq. (4.36), Eq. (4.54) is equivalent to

$$X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | X^{(\mathsf{k})}), \quad \forall \alpha \in \mathcal{A}. \tag{4.58}$$

Similarly, from Eq. (4.30), Eq. (4.55) is equivalent to

$$X^{\alpha(\mathsf{k}+1)} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} G^\alpha(X^\alpha | X^{(\mathsf{k})}), \quad \forall \alpha \in \mathcal{A}. \tag{4.59}$$

Note that both Eqs. (4.58) and (4.59) can be independently solved within a single node $\alpha \in \mathcal{A}$. Recalling $H^\alpha(X^\alpha|X^{(\mathsf{k})}) = \sum_{i=1}^{n_\alpha} H_i^\alpha(X_i^\alpha|X^{(\mathsf{k})})$ from Eq. (4.37), we conclude that Eq. (4.58) can be further reduced to $n = \sum_{\alpha\in\mathcal{A}} n_\alpha$ independent optimization problems on $X_i^\alpha \in SE(d)$

$$X_i^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X_i^\alpha \in SE(d)} H_i^\alpha(X_i^\alpha | X^{(\mathsf{k})}), \qquad \forall \alpha \in \mathcal{A} \text{ and } i \in \{1, \cdots, n_\alpha\}. \tag{4.60}$$

In particular, as is shown in Section 4.7.3, Eq. (4.60) admits a closed-form solution that only involves matrix multiplication and singular value decomposition [**130**].

As a result of Eqs. (4.56) and (4.57), we conclude that iteratively minimizing $H^\alpha(X^\alpha|X^{(\mathsf{k})})$ and $G^\alpha(X^\alpha|X^{(\mathsf{k})})$ improves the estimates and reduces the objective values. In our previous works, we have shown that Eq. (4.58) can be exactly and efficiently solved using Eq. (4.60) [**59**], and a local instead of global optimal solution to Eq. (4.59) is sufficient to guarantee the convergence of distributed PGO [**1**]. Nevertheless, Eq. (4.58) fails to make full use of the local information within a single node and might induce more iterations if there are nodes with more than one poses, whereas Eq. (4.59) can still be time-consuming to find a local optimal solution and thus, restricts the performance of distributed PGO.

To address these issues, we propose a novel update rule exploiting both Eqs. (4.58) and (4.59) to enhance the overall computational efficiency, where we precompute an initial

estimate from Eq. (4.58) and then refine the initial estimate with Eq. (4.59) to get the final estimate.

4.7.2. Algorithm

Algorithm 9 The MM–PGO Method

Input: An initial iterate $X^{(0)} \in \mathcal{X}$ and $\zeta \geq \xi \geq 0$.

Output: A sequence of iterates $\{X^{(\mathsf{k})}\}$ and $\{X^{(\mathsf{k}+\frac{1}{2})}\}$.

for $\mathsf{k} \leftarrow 0, 1, 2, \cdots$ **do**

  for node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**

    retrieve $X^{\beta(\mathsf{k})}$ from $\beta \in \mathcal{N}_\alpha$

    evaluate $\omega_{ij}^{\alpha\beta(\mathsf{k})}$ and $\omega_{ji}^{\beta\alpha(\mathsf{k})}$ using Eq. (4.19)

    evaluate $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$ using Eq. (4.43)

    $X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | X^{(\mathsf{k})})$ using Algorithm 10

    $X^{\alpha(\mathsf{k}+1)} \leftarrow$ improve $\arg\min_{X^\alpha \in \mathcal{X}^\alpha} G^\alpha(X^\alpha | X^{(\mathsf{k})})$ with $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ as the initial guess

  end for

end for

Algorithm 10 Solve $X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | X^{(\mathsf{k})})$

Input: $X^{\alpha(\mathsf{k})}$ and $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$.

Output: $X^{\alpha(\mathsf{k}+\frac{1}{2})}$.

for $i \leftarrow 1, \cdots, n_\alpha$ **do**

  $X_i^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X_i^\alpha \in SE(d)} H_i^\alpha(X_i^\alpha | X^{(\mathsf{k})})$ using Section 4.7.3

end for

retrieve $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ from $X_i^{\alpha(\mathsf{k}+\frac{1}{2})}$ where $i = 1, \cdots, n_\alpha$

The proposed update rule results in the MM–PGO method for distributed PGO (Algorithm 9). The outline of the MM–PGO method is as follows:

1) In line 5 of Algorithm 9, each node α performs one inter-node communication round to retrieve $X^{\beta(\mathsf{k})}$ from its neighbors $\beta \in \mathcal{N}_\alpha$. We remark that no other inter-node communication is required.
2) In lines 6, 7 of Algorithm 9, each node α evaluates $\omega_{ij}^{\alpha\beta(\mathsf{k})}$, $\omega_{ji}^{\beta\alpha(\mathsf{k})}$, $\nabla_{X^\alpha} F(X)$ using $X^{\alpha(\mathsf{k})}$ and $X^{\beta(\mathsf{k})}$ where $\beta \in \mathcal{N}^\alpha$ are the neighbors of node α.
3) In line 8 of Algorithm 9, we obtain the intermediate solution $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ using Algorithm 10. We have proved that the resulting $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ is already sufficient to guarantee the convergence to first-order critical points.
4) In line 3 of Algorithm 10, there exists an exact and efficient closed-form solution to $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ using Section 4.7.3.
5) In line 9 of Algorithm 9, we use $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ to initialize Eq. (4.59), and improve the final solution $X^{\alpha(\mathsf{k}+1)}$ through iterative optimization such that $G^\alpha(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) \leq G^\alpha(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})})$. Note that $X^{\alpha(\mathsf{k}+1)}$ does not have to be a local optimal solution to Eq. (4.59), nevertheless, $X^{\alpha(\mathsf{k}+1)}$ is still expected to have a faster convergence than $X^{\alpha(\mathsf{k}+\frac{1}{2})}$.

Remark 4.4. The MM–PGO method requires no inter-node communication except for lines 6 and 7 of Algorithm 9 that evaluate $\omega_{ij}^{\alpha\beta(\mathsf{k})}$, $\omega_{ji}^{\beta\alpha(\mathsf{k})}$ and $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$ using Eqs. (4.19) and (4.43), which, as mentioned before, can be distributed with limited local communication between neighboring nodes α and β without introducing any additional computation.

Since $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ in Eq. (4.58) has a closed-form solution that can be efficiently computed, and Eq. (4.59) does not require $X^{\alpha(\mathsf{k}+1)}$ to be a local optimal solution, the overall computational efficiency of the MM–PGO method is significantly improved in contrast to [**1**, **59**]. More importantly, the MM–PGO method still converges to first-order critical points as long as the following assumption holds.

Assumption 4.3. For $X^{\alpha(\mathsf{k}+1)}$ and $X^{\alpha(\mathsf{k}+\frac{1}{2})}$, it is assumed that

$$G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) \leq G^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \tag{4.61}$$

for each node $\alpha = 1, 2 \cdots, |\mathcal{A}|$.

It is known from Proposition 4.6.3 that $H^{\alpha}(X^{\alpha}|X^{(\mathsf{k})}) \geq G^{\alpha}(X^{\alpha}|X^{(\mathsf{k})})$ and $H^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}) = 0$, and thus, Assumption 4.3 can be satisfied with ease as long as line 9 of Algorithm 9 is initialized with $X^{\alpha(\mathsf{k}+\frac{1}{2})}$. Then, the MM–PGO method in Algorithm 9 is guaranteed to converge to first-order critical points as the following proposition states.

Proposition 4.7.1. For a sequence of $\{X^{(\mathsf{k})}\}$ generated by the MM–PGO method in Algorithm 9, we have the following results if Assumptions 4.1 to 4.3 hold:

(a) $F(X^{(\mathsf{k})})$ is nonincreasing as $\mathsf{k} \to \infty$;

(b) $F(X^{(\mathsf{k})}) \to F^{\infty}$ as $\mathsf{k} \to \infty$;

(c) $\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0$ as $\mathsf{k} \to \infty$ if $\xi > 0$;

(d) $\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \to 0$ as $\mathsf{k} \to \infty$ if $\zeta > \xi > 0$;

(e) if $\zeta > \xi > 0$, then there exists $\epsilon > 0$ such that

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

for any $\mathsf{K} \geq 0$;

(f) if $\zeta > \xi > 0$, then $\operatorname{grad} F(X^{(\mathsf{k})}) \to \mathbf{0}$ and $\operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})}) \to \mathbf{0}$ as $\mathsf{k} \to \infty$.

Proof. See Section 4.12.4 □

The MM–PGO method is guaranteed to converge as long as $\zeta > \xi > 0$. In contrast to other distributed PGO algorithms that are presented in [**5**, **6**, **123**, **124**], the MM–PGO method needs milder conditions and less communication to guarantee the convergence while applying to a broader class of loss kernels for distributed PGO.

4.7.3. The Closed-Form Solution to Eq. (4.60)

Substituting Eq. (4.49) into Eq. (4.60) and rewriting the resulting equation in terms of $X_i^\alpha = \begin{bmatrix} t_i^\alpha & R_i^\alpha \end{bmatrix} \in \mathbb{R}^{d\times(d+1)}$ leads to

$$\begin{aligned}\big(t_i^{\alpha(\mathsf{k}+\frac{1}{2})},\, R_i^{\alpha(\mathsf{k}+\frac{1}{2})}\big) = \arg\min_{t_i^\alpha\in\mathbb{R}^d,\, R_i^\alpha\in SO(d)} \; & \frac{1}{2}\|t_i^\alpha - t_i^{\alpha(\mathsf{k})}\|^2_{\Pi_i^{\tau,\alpha(\mathsf{k})}} + \\ & \Pi_i^{\nu,\alpha(\mathsf{k})}(R_i^\alpha - R_i^{\alpha(\mathsf{k})})^\top(t_i^\alpha - t_i^{\alpha(\mathsf{k})}) + \frac{1}{2}\|R_i^\alpha - R_i^{\alpha(\mathsf{k})}\|^2_{\Pi_i^{\kappa,\alpha(\mathsf{k})}} + \\ & \big\langle \nabla_{t_i^\alpha} F(X^{(\mathsf{k})}), t_i^\alpha - t_i^{\alpha(\mathsf{k})}\big\rangle + \big\langle \nabla_{R_i^\alpha} F(X^{(\mathsf{k})}), R_i^\alpha - R_i^{\alpha(\mathsf{k})}\big\rangle.\end{aligned}$$

For notational simplicity, the equation above is simplified to

$$\begin{aligned}\min_{t_i^\alpha\in\mathbb{R}^d,\, R_i^\alpha\in SO(d)} \; & \frac{1}{2}\|t_i^\alpha\|^2_{\Pi_i^{\tau,\alpha(\mathsf{k})}} + \Pi_i^{\nu,\alpha(\mathsf{k})} R_i^{\alpha\top} t_i^\alpha + \\ & \frac{1}{2}\|R_i^\alpha\|^2_{\Pi_i^{\kappa,\alpha(\mathsf{k})}} + \big\langle \gamma_i^{\tau,\alpha(\mathsf{k})}, t_i^\alpha\big\rangle + \big\langle \gamma_i^{\kappa,\alpha(\mathsf{k})}, R_i^\alpha\big\rangle,\end{aligned} \tag{4.62}$$

where

$$\gamma_i^{\tau,\alpha(\mathsf{k})} = \nabla_{t_i^\alpha} F(X^{(\mathsf{k})}) - t_i^{\alpha(\mathsf{k})} \Pi_i^{\tau,\alpha(\mathsf{k})} \in \mathbb{R}^d \tag{4.63}$$

and

$$\gamma_i^{\kappa,\alpha(\mathsf{k})} = \nabla_{R_i^\alpha} F(X^{(\mathsf{k})}) - R_i^{\alpha(\mathsf{k})} \Pi_i^{\kappa,\alpha(\mathsf{k})} \in \mathbb{R}^{\mathrm{d}\times d}. \tag{4.64}$$

Recalling from Eq. (4.49) that $\Pi_i^{\tau,\alpha(\mathsf{k})} \in \mathbb{R}$ and $\Pi_i^{\tau,\alpha(\mathsf{k})} > 0$, we obtain that

$$t_i^\alpha = -R_i^\alpha \Pi_i^{\nu,\alpha(\mathsf{k})^\top} \Pi_i^{\tau,\alpha(\mathsf{k})^{-1}} - \gamma_i^{\tau,\alpha(\mathsf{k})} \Pi_i^{\tau,\alpha(\mathsf{k})^{-1}} \tag{4.65}$$

minimizes Eq. (4.62) if $R_i^\alpha \in SO(d)$ is given. Then, substituting Eq. (4.65) into Eq. (4.62) yields

$$R_i^{\alpha(\mathsf{k}+\frac{1}{2})} = \arg\min_{R_i^\alpha \in SO(d)} \frac{1}{2} \|R_i^\alpha\|^2_{\Xi_i^{\alpha(\mathsf{k})}} - \left\langle \upsilon_i^{\alpha(\mathsf{k})}, R_i^\alpha \right\rangle, \tag{4.66}$$

where

$$\Xi_i^{\alpha(\mathsf{k})} = \Pi_i^{\kappa,\alpha(\mathsf{k})} - \Pi_i^{\nu,\alpha(\mathsf{k})^\top} \Pi_i^{\tau,\alpha(\mathsf{k})^{-1}} \Pi_i^{\nu,\alpha(\mathsf{k})} \in \mathbb{R}^{d\times d} \tag{4.67}$$

and

$$\upsilon_i^{\alpha(\mathsf{k})} = \gamma_i^{\tau,\alpha(\mathsf{k})} \Pi_i^{\tau,\alpha(\mathsf{k})^{-1}} \Pi_i^{\nu,\alpha(\mathsf{k})} - \gamma_i^{\kappa,\alpha(\mathsf{k})} \in \mathbb{R}^{d\times d}. \tag{4.68}$$

If we apply $R_i^{\alpha\top} R_i^\alpha = \mathbf{I}$ on Eq. (4.66), then Eq. (4.60) is equivalent to

$$R_i^{\alpha(\mathsf{k}+\frac{1}{2})} = \arg\max_{R_i^\alpha \in SO(d)} \left\langle \upsilon_i^{\alpha(\mathsf{k})}, R_i^\alpha \right\rangle. \tag{4.69}$$

Thus, Eq. (4.60) is reduced to an optimization problem on $SO(d)$, which has a closed-form solution as follows.

Following [**130**], if $\upsilon_i^{\alpha(\mathsf{k})} \in \mathbb{R}^{d\times d}$ admits a singular value decomposition

$$\upsilon_i^{\alpha(\mathsf{k})} = U_i^{\alpha(\mathsf{k})}\Sigma_i^{\alpha(\mathsf{k})}{V_i^{\alpha(\mathsf{k})}}^\top$$

where $U_i^{\alpha(\mathsf{k})}$ and $V_i^{\alpha(\mathsf{k})} \in O(d)$ are orthogonal (not necessarily special orthogonal) matrices, and $\Sigma_i^{\alpha(\mathsf{k})} = \mathrm{diag}\{\sigma_1^{\alpha(\mathsf{k})}, \sigma_2^{\alpha(\mathsf{k})}, \cdots, \sigma_d^{\alpha(\mathsf{k})}\} \in \mathbb{R}^{d\times d}$ is a diagonal matrix, and $\sigma_1^{\alpha(\mathsf{k})} \geq \sigma_2^{\alpha(\mathsf{k})} \geq \cdots \geq \sigma_d^{\alpha(\mathsf{k})} \geq 0$ are singular values of $\upsilon_i^{\alpha(\mathsf{k})}$, then the optimal solution to Eq. (4.69) is

$$R_i^{\alpha(\mathsf{k}+\frac{1}{2})} = \begin{cases} U_i^{\alpha(\mathsf{k})}\Lambda^+{V_i^{\alpha(\mathsf{k})}}^\top, & \det\left(U_i^{\alpha(\mathsf{k})}{V_i^{\alpha(\mathsf{k})}}^\top\right) > 0, \\ U_i^{\alpha(\mathsf{k})}\Lambda^-{V_i^{\alpha(\mathsf{k})}}^\top, & \text{otherwise}, \end{cases} \tag{4.70}$$

where $\Lambda^+ = \mathrm{diag}\{1, 1, \cdots, 1\} \in \mathbb{R}^{d\times d}$ and $\Lambda^- = \mathrm{diag}\{1, 1, \cdots, -1\} \in \mathbb{R}^{d\times d}$. If $d = 2$, the equation above is equivalent to the polar decomposition of 2×2 matrices, and if $d = 3$, there are fast algorithms for singular value decomposition of 3×3 matrices [**131**]. As a result, Eq. (4.69) can be efficiently solved in the case of $SO(2)$ and $SO(3)$, both of which are commonly used in SLAM.

As long as $R_i^{\alpha(\mathsf{k}+\frac{1}{2})} \in SO(d)$ is known, $t_i^{\alpha(\mathsf{k}+\frac{1}{2})} \in \mathbb{R}^d$ can be exactly recovered using Eq. (4.65):

$$t_i^{\alpha(\mathsf{k}+\frac{1}{2})} = -R_i^{\alpha(\mathsf{k}+\frac{1}{2})}{\Pi_i^{\nu,\alpha(\mathsf{k})}}^\top{\Pi_i^{\tau,\alpha(\mathsf{k})}}^{-1} - \gamma_i^{\tau,\alpha(\mathsf{k})}{\Pi_i^{\tau,\alpha(\mathsf{k})}}^{-1}. \tag{4.71}$$

Therefore, $X_i^{\alpha(\mathsf{k}+\frac{1}{2})} = \begin{bmatrix} t_i^{\alpha(\mathsf{k}+\frac{1}{2})} & R_i^{\alpha(\mathsf{k}+\frac{1}{2})} \end{bmatrix} \in \mathbb{R}^{d\times(d+1)}$ is exactly solved, whose computation only involves matrix multiplication and singular value decomposition.

4.8. The Accelerated Majorization Minimization Method for Distributed Pose Graph Optimization with a Master Node

In the last several decades, a number of accelerated first-order optimization methods have been proposed [**120**, **121**]. Even though most of them were originally developed for convex optimization, it has been recently found that these accelerated methods have a good performance for nonconvex optimization as well [**132**–**134**]. In our previous works [**1**, **59**], we proposed to use Nesterov's method to accelerate first-order optimization methods for distributed PGO, which empirically have much faster convergence. Since the MM–PGO method is a first-order optimization method, it is possible to exploit Nesterov's method for acceleration as well.

In this and next sections, we will propose the accelerated MM methods for distributed PGO with and without a master node, respectively, both of which significantly improve the convergence compared to MM–PGO.

4.8.1. Nesterov's Method

From Eqs. (4.29) and (4.35), it is obvious that $G(X|X^{(\mathsf{k})})$ and $H(X|X^{(\mathsf{k})})$ are proximal operators of $F(X)$, which suggests the possibility to use Nesterov's methods [**120**, **121**] for acceleration.

As a result of Nesterov's method [**120**, **121**], $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ and $X^{\alpha(\mathsf{k}+1)}$ can be updated with

$$s^{\alpha(\mathsf{k}+1)} = \frac{\sqrt{4{s^{\alpha(\mathsf{k})}}^2 + 1} + 1}{2}, \tag{4.72}$$

$$\lambda^{\alpha(\mathsf{k})} = \frac{s^{\alpha(\mathsf{k})} - 1}{s^{\alpha(\mathsf{k}+1)}}, \tag{4.73}$$

$$Y^{\alpha(\mathsf{k})} = X^{\alpha(\mathsf{k})} + \lambda^{\alpha(\mathsf{k})} \cdot \left(X^{\alpha(\mathsf{k})} - X^{\alpha(\mathsf{k}-1)}\right), \tag{4.74}$$

$$X^{\alpha(\mathsf{k}+\frac{1}{2})} = \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | Y^{(\mathsf{k})}), \tag{4.75}$$

$$X^{\alpha(\mathsf{k}+1)} = \arg\min_{X^\alpha \in \mathcal{X}^\alpha} G^\alpha(X^\alpha | Y^{(\mathsf{k})}). \tag{4.76}$$

In Eqs. (4.75) and (4.76), $G^\alpha(\cdot|Y^{(\mathsf{k})}) : \mathcal{X}^\alpha \to \mathbb{R}$ and $H^\alpha(\cdot|Y^{(\mathsf{k})}) : \mathcal{X}^\alpha \to \mathbb{R}$ are proximal operators defined as:

$$G^\alpha(X^\alpha | Y^{(\mathsf{k})}) = \frac{1}{2}\|X^\alpha - Y^{\alpha(\mathsf{k})}\|^2_{\Gamma^{\alpha(\mathsf{k})}} + \left\langle \nabla_{X^\alpha} F(Y^{(\mathsf{k})}),\, X^\alpha - Y^{\alpha(\mathsf{k})} \right\rangle \tag{4.77}$$

and

$$H^\alpha(X^\alpha | Y^{(\mathsf{k})}) = \frac{1}{2}\|X^\alpha - Y^{\alpha(\mathsf{k})}\|^2_{\Pi^{\alpha(\mathsf{k})}} + \left\langle \nabla_{X^\alpha} F(Y^{(\mathsf{k})}),\, X^\alpha - Y^{\alpha(\mathsf{k})} \right\rangle, \tag{4.78}$$

where $\Gamma^{\alpha(\mathsf{k})}$ and $\Pi^{\alpha(\mathsf{k})}$ are the same as these in $G^\alpha(\cdot|X^{(\mathsf{k})})$ and $H^\alpha(\cdot|X^{(\mathsf{k})})$ in Eqs. (4.31) and (4.38).

The key idea of Nesterov's method is to exploit the momentum $X^{\alpha(\mathsf{k})} - X^{\alpha(\mathsf{k}-1)}$ for acceleration, which is essentially governed by Eqs. (4.72) to (4.74). Note that Nesterov's method using Eqs. (4.72) to (4.76) suggest is equivalent to Eqs. (4.58) and (4.59) when $s^{\alpha(\mathsf{k})} = 1$ and $\lambda^{\alpha(\mathsf{k})} = 0$, and then increasingly affected by the momentum as $s^{\alpha(\mathsf{k})}$ and $\lambda^{\alpha(\mathsf{k})}$ increase. Similar to **[1, 59]**, the implementation of Nesterov's method **[120, 121]** in practice leads to significant speedup of distributed PGO. Moreover, Eqs. (4.72) to (4.78) can be distributed without introducing any additional computation as long as each node

$\alpha \in \mathcal{A}$ can communicate with its neighbors $\beta \in \mathcal{N}^\alpha$. Thus, it is preferable to adopt Nesterov's method to accelerate distributed PGO.

4.8.2. Adaptive Restart

In spite of faster convergence, Nesterov's accelerated distributed PGO using Eqs. (4.72) to (4.76) is no longer nonincreasing, and might fail to converge due to the nonconvexity of PGO. Fortunately, such a problem can be remedied with an adaptive restart scheme [**1**, **59**, **122**].

In the adaptive restart scheme , we recursively define $\overline{F}^{(\mathsf{k})}$ that is an exponential moving averaging of $F(X^{(0)})$, $F(X^{(1)})$, $\cdots$, $F(X^{(\mathsf{k})})$ [**59**, **134**, **135**]:

$$\overline{F}^{(\mathsf{k})} \triangleq \begin{cases} F(X^{(0)}), & \mathsf{k} = 0, \\ (1-\eta) \cdot \overline{F}^{(\mathsf{k}-1)} + \eta \cdot F(X^{(\mathsf{k})}), & \text{otherwise} \end{cases} \tag{4.79}$$

where $\eta \in (0,\, 1]$. Then, $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ and $X^{(\mathsf{k}+1)}$ are updated using the following steps:

(1) Update $X^{(\mathsf{k}+\frac{1}{2})}$ and $X^{(\mathsf{k}+1)}$ by solving Eqs. (4.75) and (4.76) with $Y^{(\mathsf{k})}$ resulting from Eqs. (4.72) to (4.74);

(2) If $F(X^{(\mathsf{k}+\frac{1}{2})}) > \overline{F}^{(\mathsf{k})}$, update $X^{(\mathsf{k}+\frac{1}{2})}$ again by solving Eq. (4.75) with $Y^{(\mathsf{k})} = X^{(\mathsf{k})}$;

(3) If $F(X^{(\mathsf{k}+1)}) > \overline{F}^{(\mathsf{k})}$, update $X^{(\mathsf{k}+1)}$ again by solving Eq. (4.76) with $Y^{(\mathsf{k})} = X^{(\mathsf{k})}$ and reduce $s^{\alpha(\mathsf{k}+1))}$.

Remark 4.5. Since $\eta \in (0,\, 1]$ in Eq. (4.79), $\overline{F}^{(\mathsf{k}+1)} \leq \overline{F}^{(\mathsf{k})}$ as long as $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$. In Section 4.12.5, we have proved $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$ if $X^{(\mathsf{k}+\frac{1}{2})}$ and $X^{(\mathsf{k}+1)}$ are updated with $Y^{(\mathsf{k})} = X^{(\mathsf{k})}$ in Eqs. (4.75) and (4.76). Therefore, such an adaptive restart scheme is guaranteed to result in a nonincreasing sequence of $\overline{F}^{(\mathsf{k})}$.

Note that one has to aggregate information across the network to evaluate $\overline{F}^{(\mathsf{k})}$, $F(X^{(\mathsf{k}+\frac{1}{2})})$, $F(X^{(\mathsf{k}+1)})$ using Eqs. (4.12) and (4.79) and a master node capable of communicating with each node $\alpha \in \mathcal{A}$ is required. Thus, we make the following assumption about the existence of such a master node in the rest of this section.

Assumption 4.4. There is a master node to retrieve $X^{\alpha(\mathsf{k})}$ and $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ from each node $\alpha \in \mathcal{A}$ and evaluate $\overline{F}^{(\mathsf{k})}$, $F(X^{(\mathsf{k}+\frac{1}{2})})$, $F(X^{(\mathsf{k}+1)})$.

aster node to collect $X^{\alpha(\mathsf{k})}$ and $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ from each node $\alpha \in \mathcal{A}$ to evaluate $F(X^{(\mathsf{k})})$, $F(X^{(\mathsf{k}+\frac{1}{2})})$ and $\overline{F}^{(\mathsf{k})}$.

4.8.3. Algorithm

From Nesterov's method and the adaptive restart scheme in Eqs. (4.72) to (4.76) and (4.79), we obtain the AMM–PGO* method for distributed PGO (Algorithm 11), where "$*$" indicates that there is a master node for the distributed PGO.

The outline of the AMM–PGO* method is as follows:

1) In lines 11, 12 of Algorithm 11, each node α computes $Y^{(\mathsf{k})}$ for Nesterov's acceleration that is related with $s^{\alpha(\mathsf{k})} \in [1, \infty)$ and $\lambda^{\alpha(\mathsf{k})} \in [0, 1)$.
2) In line 2 of Algorithm 12, each node α performs one inter-node communication round to retrieve $X^{\beta(\mathsf{k})}$ and $Y^{\beta(\mathsf{k})}$ from its neighbors $\beta \in \mathcal{N}^\alpha$.
3) In lines 7, 13, 21 of Algorithm 12, each node α performs one inter-node communication round to send $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ and $X^{\alpha(\mathsf{k}+1)}$ to the master node.

Algorithm 11 The AMM–PGO* Method

1: **Input**: An initial iterate $X^{(0)} \in \mathcal{X}$, and $\zeta \geq \xi \geq 0$, and $\eta \in (0, 1]$, and $\psi > 0$, and $\phi > 0$.
2: **Output**: A sequence of iterates $\{X^{(\mathsf{k})}\}$ and $\{X^{(\mathsf{k}+\frac{1}{2})}\}$.
3: **for** node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**
4: $\quad X^{\alpha(-1)} \leftarrow X^{\alpha(0)}$ and $s^{\alpha(0)} \leftarrow 1$
5: $\quad$ send $X^{\alpha(0)}$ to the master node
6: **end for**
7: evaluate $F(X^{(0)})$ using Eq. (4.11) at the master node
8: $\overline{F}^{(-1)} \leftarrow F(X^{(0)})$ at the master node
9: **for** $\mathsf{k} \leftarrow 0, 1, 2, \cdots$ **do**
10: $\quad$ **for** node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**
11: $\quad\quad s^{\alpha(\mathsf{k}+1)} \leftarrow \frac{\sqrt{4{s^{\alpha(\mathsf{k})}}^2+1}+1}{2}$, $\lambda^{\alpha(\mathsf{k})} \leftarrow \frac{s^{\alpha(\mathsf{k})}-1}{s^{\alpha(\mathsf{k}+1)}}$
12: $\quad\quad Y^{\alpha(\mathsf{k})} \leftarrow X^{\alpha(\mathsf{k})} + \lambda^{\alpha(\mathsf{k})} \cdot \left(X^{\alpha(\mathsf{k})} - X^{\alpha(\mathsf{k}-1)}\right)$
13: $\quad$ **end for**
14: $\quad \overline{F}^{(\mathsf{k})} \leftarrow (1-\eta) \cdot \overline{F}^{(\mathsf{k}-1)} + \eta \cdot F(X^{(\mathsf{k})})$ at the master node
15: $\quad$ update $X^{(\mathsf{k}+\frac{1}{2})}$ and $X^{(\mathsf{k}+1)}$ using Algorithm 12
16: **end for**

4) In lines 3, 4 of Algorithm 12, each node α evaluates $\omega_{ij}^{\alpha\beta(\mathsf{k})}$, $\omega_{ji}^{\beta\alpha(\mathsf{k})}$, $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$, $\nabla_{X^\alpha} F(Y^{(\mathsf{k})})$ using $X^{\alpha(\mathsf{k})}$, $Y^{\alpha(\mathsf{k})}$, $X^{\beta(\mathsf{k})}$, $Y^{\beta(\mathsf{k})}$ where $\beta \in \mathcal{N}^\alpha$ are the neighbors of node α.

5) In lines 8, 14 of Algorithm 11 and lines 9, 15, 23 of Algorithm 12, the master node evaluates $\overline{F}^{(\mathsf{k})}$, $F(X^{(\mathsf{k}+\frac{1}{2})})$, $F(X^{(\mathsf{k}+1)})$ that are used for adaptive restart.

6) In lines 10 to 24 of Algorithm 12, the master node performs adaptive restart to keep $F(X^{(\mathsf{k}+\frac{1}{2})}) \leq \overline{F}^{(\mathsf{k})}$ and $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$, which yields a nonincreasing sequence of $\overline{F}^{(\mathsf{k})}$ to guarantee the convergence.

7) In lines 6, 19 of Algorithm 12, note that $X^{\alpha(\mathsf{k}+1)}$ does not have to be a local optimal solution to Eq. (4.59).

Algorithm 12 Updates for the AMM–PGO* Method

1: **for** node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**
2: retrieve $X^{\beta(\mathsf{k})}$ and $Y^{\beta(\mathsf{k})}$ from $\beta \in \mathcal{N}_\alpha$
3: evaluate $\omega_{ij}^{\alpha\beta(\mathsf{k})}$ and $\omega_{ji}^{\beta\alpha(\mathsf{k})}$ using Eq. (4.19)
4: evaluate $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$ and $\nabla_{X^\alpha} F(Y^{(\mathsf{k})})$ using Eq. (4.43)
5: $X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | Y^{(\mathsf{k})})$ using Algorithm 10
6: $X^{\alpha(\mathsf{k}+1)} \leftarrow$ improve $\arg\min_{X^\alpha \in \mathcal{X}^\alpha} G^\alpha(X^\alpha | Y^{(\mathsf{k})})$ with $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ as the initial guess
7: send $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ and $X^{\alpha(\mathsf{k}+1)}$ to the master node
8: **end for**
9: evaluate $F(X^{(\mathsf{k}+\frac{1}{2})})$ and $F(X^{(\mathsf{k}+1)})$ using Eq. (4.11) at the master node
10: **if** $F(X^{(\mathsf{k}+\frac{1}{2})}) > \overline{F}^{(\mathsf{k})} - \psi \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2$ **then**
11: **for** node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**
12: $X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | X^{(\mathsf{k})})$ using Algorithm 10
13: send $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ to the master node
14: **end for**
15: evaluate $F(X^{(\mathsf{k}+\frac{1}{2})})$ using Eq. (4.11) at the master node
16: **end if**
17: **if** $F(X^{(\mathsf{k}+1)}) > \overline{F}^{(\mathsf{k})} - \psi \cdot \|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2$ **then**
18: **for** node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**
19: $X^{\alpha(\mathsf{k}+1)} \leftarrow$ improve $\arg\min_{X^\alpha \in \mathcal{X}^\alpha} G^\alpha(X^\alpha | X^{(\mathsf{k})})$ with $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ as the initial guess
20: $s^{\alpha(\mathsf{k}+1)} \leftarrow \max\{\frac{1}{2} s^{\alpha(\mathsf{k}+1)}, 1\}$
21: send $X^{\alpha(\mathsf{k}+1)}$ to the master node
22: **end for**
23: evaluate $F(X^{(\mathsf{k}+1)})$ using Eq. (4.11) at the master node
24: **end if**
25: **if** $\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) < \phi \cdot \left(\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+\frac{1}{2})})\right)$ **then**
26: $X^{(\mathsf{k}+1)} \leftarrow X^{(\mathsf{k}+\frac{1}{2})}$ and $F(X^{(\mathsf{k}+1)}) \leftarrow F(X^{(\mathsf{k}+\frac{1}{2})})$
27: **end if**

8) In lines 25 to 27 of Algorithm 12, $F(X^{(\mathsf{k}+1)})$ is guaranteed to yield sufficient improvement over $\overline{F}^{(\mathsf{k})}$ compared to $F(X^{(\mathsf{k}+\frac{1}{2})})$.

In spite of acceleration, the AMM–PGO* method is guaranteed to converge to first-order critical points under mild conditions as the following proposition states.

Proposition 4.8.1. If Assumptions 4.1 to 4.4 hold, then for a sequence of iterates $\{X^{(\mathsf{k})}\}$ generated by Algorithm 11, we obtain

(a) $\overline{F}^{(k)}$ is nonincreasing;

(b) $F(X^{(\mathsf{k})}) \to F^\infty$ and $\overline{F}^{(k)} \to F^\infty$ as $\mathsf{k} \to \infty$;

(c) $\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0$ as $\mathsf{k} \to \infty$ if $\xi > 0$ and $\zeta > 0$;

(d) $\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \to 0$ as $\mathsf{k} \to \infty$ if $\zeta \geq \xi > 0$;

(e) if $\zeta \geq \xi > 0$, then there exists $\epsilon > 0$ such that

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

for any $\mathsf{K} \geq 0$;

(f) if $\zeta > \xi > 0$, then $\mathrm{grad}\, F(X^{(\mathsf{k})}) \to \mathbf{0}$ and $\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})}) \to \mathbf{0}$ as $\mathsf{k} \to \infty$.

Proof. See Section 4.12.5. □

Remark 4.6. If $\eta = 1$ in Eq. (4.79), $F(X^{(\mathsf{k})}) = \overline{F}^{(\mathsf{k})}$, and $F(X^{(\mathsf{k})})$ is also nonincreasing according to Proposition 4.8.1(a). Even though $F(X^{(\mathsf{k})})$ might fail to be nonincreasing, we still recommend to choose $\eta \ll 1$ that empirically yields fewer adaptive restarts and faster convergence for distributed PGO.

Remark 4.7. In Algorithm 12, $\psi > 0$ and $\phi > 0$ guarantee that $F(X^{(\mathsf{k}+\frac{1}{2})})$ and $F(X^{(\mathsf{k}+1)})$ yield sufficient improvement over $\overline{F}^{(\mathsf{k})}$ in terms of $\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|$ and $\|X^{(\mathsf{k}+1)} -$

$X^{(\mathsf{k})}\|$, and are recommended to set close to zero to avoid unnecessary adaptive restarts and make better use of Nesterov's acceleration.

4.9. The Accelerated Majorization Minimization Method for Distributed Pose Graph Optimization without a Master Node

In this section, we will propose accelerated MM methods for distributed PGO without a master node. The resulting accelerated MM method not only is guaranteed to converge to first-order critical points with limited local communication but also has almost no loss of computational efficiency in contrast to the AMM–PGO* method with a master node (see Section 4.10).

4.9.1. Adaptive Restart Scheme

The adaptive restart is essential for the convergence of accelerated MM methods. In the AMM–PGO* method (Algorithm 11), the adaptive restart scheme needs a master node to evaluate $F(X^{(\mathsf{k})})$ and $\overline{F}^{(\mathsf{k})}$ and guarantee the convergence. However, in the case of no master node, such an adaptive restart scheme requires substantial amount of inter-node communication across the network, making the AMM–PGO* method unscalable for large-scale distributed PGO. Recently, we developed an adaptive restart scheme for distributed PGO that does not require a master node but still generates convergent iterates with limited local communication [**1**]. In spite of that, the adaptive restart scheme in [**1**] is conservative and suffers from unnecessary restarts, which hinders acceleration and yields slower convergence than the AMM–PGO* method. Thus, we need to redesign the adaptive restart scheme to boost the performance of distributed PGO without master node.

Recall that the AMM–PGO* method's adaptive restart scheme is for the purpose of $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$. In a similar way, we propose the following adaptive restart scheme that keeps $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$ for distributed PGO without master node, from which the convergence is guaranteed for accelerated MM methods.

For notational simplicity, we introduce $\Delta G^{\alpha}(X|X^{(\mathsf{k})}) : \mathcal{X} \to \mathbb{R}$:

$$\Delta G^{\alpha}(X|X^{(\mathsf{k})}) \triangleq \frac{1}{2} \sum_{\beta \in \mathcal{N}_{-}^{\alpha}} \sum_{(i,j) \in \overrightarrow{\mathcal{E}}^{\alpha\beta}} \Big(F_{ij}^{\alpha\beta}(X) - E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})}) \Big) + \frac{1}{2} \sum_{\beta \in \mathcal{N}_{+}^{\alpha}} \sum_{(i,j) \in \overrightarrow{\mathcal{E}}^{\beta\alpha}} \Big(F_{ij}^{\beta\alpha}(X) - E_{ij}^{\beta\alpha}(X|X^{(\mathsf{k})}) \Big) - \frac{\xi}{2} \big\| X^{\alpha} - X^{\alpha(\mathsf{k})} \big\|^2, \tag{4.80}$$

where $F_{ij}^{\alpha\beta}(X)$, $F_{ij}^{\beta\alpha}(X)$, $E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})})$, $E_{ij}^{\beta\alpha}(X|X^{(\mathsf{k})})$ are given in Eqs. (4.14) and (4.27). From $\Delta G^{\alpha}(X|X^{(\mathsf{k})})$ in Eq. (4.80), we recursively define $F^{\alpha(\mathsf{k})}$, $\overline{F}^{\alpha(\mathsf{k})}$, $G^{\alpha(\mathsf{k})}$ according to:

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

$$F^{\alpha(-1)} \triangleq \sum_{(i,j) \in \overrightarrow{\mathcal{E}}^{\alpha\alpha}} F_{ij}^{\alpha\alpha}(X^{(0)}) + \frac{1}{2} \sum_{\beta \in \mathcal{N}_{-}^{\alpha}} \sum_{(i,j) \in \overrightarrow{\mathcal{E}}^{\alpha\beta}} F_{ij}^{\alpha\beta}(X^{(0)}) + \frac{1}{2} \sum_{\beta \in \mathcal{N}_{+}^{\alpha}} \sum_{(i,j) \in \overrightarrow{\mathcal{E}}^{\beta\alpha}} F_{ij}^{\beta\alpha}(X^{(0)}) \tag{4.81}$$

and

$$\overline{F}^{\alpha(-1)} \triangleq F^{\alpha(-1)}. \tag{4.82}$$

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

$$G^{\alpha(\mathsf{k})} \triangleq G^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k}-1)}) + F^{\alpha(\mathsf{k}-1)}, \tag{4.83}$$

$$F^{\alpha(\mathsf{k})} \triangleq G^{\alpha(\mathsf{k})} + \Delta G^{\alpha}(X^{(\mathsf{k})}|X^{(\mathsf{k}-1)}), \tag{4.84}$$

$$\overline{F}^{\alpha(\mathsf{k})} \triangleq (1-\eta)\cdot \overline{F}^{\alpha(\mathsf{k}-1)} + \eta \cdot F^{\alpha(\mathsf{k})} \tag{4.85}$$

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

From the definitions of $F_{ij}^{\alpha\alpha}(X)$, $F_{ij}^{\alpha\beta}(X)$, $F_{ij}^{\beta\alpha}(X)$, $G^{\alpha}(X^{\alpha}|X^{(\mathsf{k})})$, $\Delta G^{\alpha}(X|X^{(\mathsf{k})})$ in Eqs. (4.14), (4.17), (4.31) and (4.80), it is tedious but straightforward to show that $G^{\alpha(\mathsf{k})}$, $F^{\alpha(\mathsf{k})}$, $\overline{F}^{\alpha(\mathsf{k})}$ in Eqs. (4.81) to (4.85) can be explicitly evaluated with one inter-node communication round between node α and its neighbors $\beta \in \mathcal{N}_{\alpha}$. Furthermore, we have the following proposition about $F^{\alpha(\mathsf{k})}$, $\overline{F}^{\alpha(\mathsf{k})}$, $G^{\alpha(\mathsf{k})}$.

Proposition 4.9.1. For any $\mathsf{k} \geq 0$, we have

(a) $F(X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k})}$ where $F(X^{(\mathsf{k})})$ is given in Eq. (4.11);

(b) $\overline{F}^{(\mathsf{k})} = \sum_{\alpha\in\mathcal{A}} \overline{F}^{\alpha(\mathsf{k})}$ where $\overline{F}^{(\mathsf{k})}$ is given in Eq. (4.79);

(c) $F^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$ if $G^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$.

Proof. See Section 4.12.6. □

In Algorithms 11 and 12, the AMM–PGO* method's adaptive restart scheme requires a master node to evaluate and compare $F(X^{(\mathsf{k}+1)})$ and $\overline{F}^{(\mathsf{k})}$ in order to keep $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$. In spite of local communication, we remark it is still possible to result in $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$ for distributed PGO without master node as follows:

(1) From Propositions 4.9.1(a) and 4.9.1(b), we obtain $F(X^{(\mathsf{k}+1)}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k}+1)}$ and $\overline{F}^{(\mathsf{k})} = \sum_{\alpha\in\mathcal{A}} \overline{F}^{\alpha(\mathsf{k})}$, and as a result,

$$F(X^{(k+1)}) = \sum_{\alpha \in \mathcal{A}} F^{\alpha(k+1)} \leq \sum_{\alpha \in \mathcal{A}} \overline{F}^{\alpha(k)} = \overline{F}^{(k)} \tag{4.86}$$

as long as

$$F^{\alpha(k+1)} \leq \overline{F}^{\alpha(k)} \tag{4.87}$$

for each node $\alpha \in \mathcal{A}$. In addition, Proposition 4.9.1(c) indicates that $G^{\alpha(k+1)} \leq \overline{F}^{\alpha(k)}$ leads to Eq. (4.87). Therefore, $F(X^{(k+1)}) \leq \overline{F}^{(k)}$ is reduced to requiring

$$G^{\alpha(k+1)} \leq \overline{F}^{\alpha(k)} \tag{4.88}$$

for each node $\alpha \in \mathcal{A}$.

(2) In terms of $G^{\alpha(k+1)}$ and $\overline{F}^{\alpha(k)}$, we obtain from Eq. (4.83) that Eq. (4.88) is equivalent to

$$G^{\alpha(k+1)} = G^{\alpha}(X^{\alpha(k+1)}|X^{(k)}) + F^{\alpha(k)} \leq \overline{F}^{\alpha(k)}. \tag{4.89}$$

From Eqs. (4.86) to (4.89), we conclude $F^{\alpha(k+1)} \leq \overline{F}^{\alpha(k)}$ and $F(X^{(k+1)}) \leq \overline{F}^{(k)}$ as long as

$$G^{\alpha}(X^{\alpha(k+1)}|X^{(k)}) \leq 0 \tag{4.90}$$

and

$$F^{\alpha(k)} \leq \overline{F}^{\alpha(k)} \tag{4.91}$$

for each node $\alpha \in \mathcal{A}$.

(3) Recall from Eq. (4.31) that $G^{\alpha}(X^{\alpha(k)}|X^{(k)}) = 0$. Then, if X^{α} in Eqs. (4.58) and (4.59) is initialized with $X^{\alpha(k)}$, the resulting $X^{\alpha(k+1)}$ has

$$G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) \leq G^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}) = 0, \tag{4.92}$$

which yields Eq. (4.90).

(4) In Section 4.12.7, it can be shown by induction that Eq. (4.91) holds for distributed PGO without master node.

From the discussion above, we conclude that Eqs. (4.90) and (4.91) can be always satisfied, and more importantly, result in $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$ for distributed PGO without master node. This suggests an adaptive restart scheme that evaluates and compares $G^{\alpha(\mathsf{k}+1)}$ and $\overline{F}^{\alpha(\mathsf{k})}$ independently at each node $\alpha \in \mathcal{A}$ to keep $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$. We emphasize that the resulting adaptive restart scheme requires no master node and evaluates neither $F(X^{(\mathsf{k}+1)})$ nor $\overline{F}^{(\mathsf{k})}$. Therefore, such an adaptive restart scheme differs from these in the $\mathsf{AMM\text{–}PGO}^{*}$ method and **[59, 134, 135]** that rely on a master node to evaluate and compare $F(X^{(\mathsf{k}+1)})$ and $\overline{F}^{(\mathsf{k})}$. In addition, the adaptive restart scheme takes at most one inter-node communication round among neighboring nodes at each iteration, which is well suited for distributed PGO without master node.

4.9.2. Algorithm

Following the adaptive restart scheme using $G^{\alpha(\mathsf{k}+1)}$ and $\overline{F}^{\alpha(\mathsf{k})}$, we obtain the $\mathsf{AMM\text{–}PGO}^{\#}$ method (Algorithm 13), where "#" indicates that no master node is needed.

The outline of the $\mathsf{AMM\text{–}PGO}^{\#}$ method is as follows:

1) In lines 5, 14 of Algorithm 13, each node α performs one inter-node communication round to retrieve $X^{\beta(\mathsf{k})}$ and $Y^{\beta(\mathsf{k})}$ from its neighbors $\beta \in \mathcal{N}^{\alpha}$. We remark that no other inter-node communication is required.

Algorithm 13 The AMM–PGO$^{\#}$ Method

1: **Input**: An initial iterate $X^{(0)} \in \mathcal{X}$, and $\eta \in (0, 1]$, and $\zeta \geq \xi \geq 0$, and $\psi > 0$, and $\phi > 0$.
2: **Output**: A sequence of iterates $\{X^{(\mathsf{k})}\}$ and $\{X^{(\mathsf{k}+\frac{1}{2})}\}$.
3: **for** node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**
4: $X^{\alpha(-1)} \leftarrow X^{\alpha(0)}$ and $s^{\alpha(0)} \leftarrow 1$
5: retrieve $X^{\beta(-1)}$ and $X^{\beta(0)}$ from $\beta \in \mathcal{N}_\alpha$
6: evaluate $F^{\alpha(-1)}$ using Eq. (4.81)
7: evaluate $\overline{F}^{\alpha(-1)}$ using Eq. (4.82)
8: $G^{\alpha(0)} \leftarrow G^\alpha(X^{\alpha(0)}|X^{(-1)}) + F^{\alpha(-1)}$
9: **end for**
10: **for** $\mathsf{k} \leftarrow 0, 1, 2, \cdots$ **do**
11: **for** node $\alpha \leftarrow 1, \cdots, |\mathcal{A}|$ **do**
12: $s^{\alpha(\mathsf{k}+1)} \leftarrow \dfrac{\sqrt{4{s^{\alpha(\mathsf{k})}}^2 + 1} + 1}{2}$, $\lambda^{\alpha(\mathsf{k})} \leftarrow \dfrac{s^{\alpha(\mathsf{k})} - 1}{s^{\alpha(\mathsf{k}+1)}}$
13: $Y^{\alpha(\mathsf{k})} \leftarrow X^{\alpha(\mathsf{k})} + \lambda^{\alpha(\mathsf{k})} \cdot \left(X^{\alpha(\mathsf{k})} - X^{\alpha(\mathsf{k}-1)}\right)$
14: retrieve $X^{\beta(\mathsf{k})}$ and $Y^{\beta(\mathsf{k})}$ from $\beta \in \mathcal{N}_\alpha$
15: $F^{\alpha(\mathsf{k})} \leftarrow G^{\alpha(\mathsf{k})} + \Delta G^\alpha(X^{(\mathsf{k})}|X^{(\mathsf{k}-1)})$ using Eqs. (4.80) and (4.83)
16: $\overline{F}^{\alpha(\mathsf{k})} \leftarrow (1-\eta) \cdot \overline{F}^{\alpha(\mathsf{k}-1)} + \eta \cdot F^{\alpha(\mathsf{k})}$
17: update $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ and $X^{\alpha(\mathsf{k}+1)}$ using Algorithm 14
18: **end for**
19: **end for**

2) In lines 6, 7, 15, 16 of Algorithm 13 and lines 4, 6, 9, 13 of Algorithm 14, each node α evaluates $F^{\alpha(\mathsf{k})}$, $\overline{F}^{\alpha(\mathsf{k})}$, $G^{\alpha(\mathsf{k}+\frac{1}{2})}$, $G^{\alpha(\mathsf{k}+1)}$ that are used for adaptive restart. Note that $X^{\beta(\mathsf{k})}$ and $X^{\beta(\mathsf{k}-1)}$ from node α's neighbors $\beta \in \mathcal{N}^\alpha$ are needed.

3) In lines 7 to 15 of Algorithm 14, each node α performs independent adaptive restart such that $G^{\alpha(\mathsf{k}+\frac{1}{2})} \leq \overline{F}^{\alpha(\mathsf{k})}$ and $G^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$, which also results in $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k})}$ and a nonincreasing sequence of $\overline{F}^{(\mathsf{k})}$ for distributed PGO without master node.

4) In lines 16 to 18 of Algorithm 14, $G^{\alpha(\mathsf{k}+1)}$ is guaranteed to yield sufficient improvement over $\overline{F}^{\alpha(\mathsf{k}+1)}$ compared to $G^{\alpha(\mathsf{k}+\frac{1}{2})}$.

Algorithm 14 Update $X^{(\mathsf{k}+\frac{1}{2})}$ and $X^{(\mathsf{k}+1)}$ for the AMM–PGO$^{\#}$ Method

evaluate $\omega_{ij}^{\alpha\beta(\mathsf{k})}$ and $\omega_{ji}^{\beta\alpha(\mathsf{k})}$ using Eq. (4.19)
evaluate $\nabla_{X^\alpha} F(X^{(\mathsf{k})})$ and $\nabla_{X^\alpha} F(Y^{(\mathsf{k})})$ using Eq. (4.43)
$X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | Y^{(\mathsf{k})})$ using Algorithm 10
$G^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow G^\alpha(X^{\alpha(\mathsf{k}+\frac{1}{2})} | X^{(\mathsf{k})}) + F^{\alpha(\mathsf{k})}$
$X^{\alpha(\mathsf{k}+1)} \leftarrow$ improve $\arg\min_{X^\alpha \in \mathcal{X}^\alpha} G^\alpha(X^\alpha | Y^{(\mathsf{k})})$ with $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ as the initial guess
$G^{\alpha(\mathsf{k}+1)} \leftarrow G^\alpha(X^{\alpha(\mathsf{k}+1)} | X^{(\mathsf{k})}) + F^{\alpha(\mathsf{k})}$
if $G^{\alpha(\mathsf{k}+\frac{1}{2})} > \overline{F}^{\alpha(\mathsf{k})} - \psi \cdot \|X^{\alpha(\mathsf{k}+\frac{1}{2})} - X^{\alpha(\mathsf{k})}\|^2$ **then**
    $X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | X^{(\mathsf{k})})$ using Algorithm 10
    $G^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow G^\alpha(X^{\alpha(\mathsf{k}+\frac{1}{2})} | X^{(\mathsf{k})}) + F^{\alpha(\mathsf{k})}$
end if
if $G^{\alpha(\mathsf{k}+1)} > \overline{F}^{\alpha(\mathsf{k})}$ **then**
    $X^{\alpha(\mathsf{k}+1)} \leftarrow$ improve $\arg\min_{X^\alpha \in \mathcal{X}^\alpha} G^\alpha(X^\alpha | X^{(\mathsf{k})})$ with $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ as the initial guess
    $G^{\alpha(\mathsf{k}+1)} \leftarrow G^\alpha(X^{\alpha(\mathsf{k}+1)} | X^{(\mathsf{k})}) + F^{\alpha(\mathsf{k})}$
    $s^{\alpha(\mathsf{k}+1)} \leftarrow \max\{\frac{1}{2} s^{\alpha(\mathsf{k}+1)}, 1\}$
end if
if $\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+1)} < \phi \cdot \left(\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+\frac{1}{2})}\right)$ **then**
    $X^{\alpha(\mathsf{k}+1)} \leftarrow X^{\alpha(\mathsf{k}+\frac{1}{2})}$ and $G^{\alpha(\mathsf{k}+1)} \leftarrow G^{\alpha(\mathsf{k}+\frac{1}{2})}$
end if

The AMM–PGO$^{\#}$ method is guaranteed to converge to first-order critical points under mild conditions as the following propositions states.

Proposition 4.9.2. If Assumptions 4.1 to 4.3 hold, then for a sequence of iterates $\{X^{(\mathsf{k})}\}$ generated by Algorithm 13, we obtain

(a) $\overline{F}^{(\mathsf{k})}$ is nonincreasing;

(b) $F(X^{(\mathsf{k})}) \to F^\infty$ and $\overline{F}^{(k)} \to F^\infty$ as $\mathsf{k} \to \infty$;

(c) $\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0$ as $\mathsf{k} \to \infty$ if $\zeta > \xi > 0$;

(d) $\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \to 0$ as $\mathsf{k} \to \infty$ if $\zeta > \xi > 0$;

(e) if $\zeta > \xi > 0$, then there exists $\epsilon > 0$ such that

$$\min_{0 \le \mathsf{k} < \mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\| \le 2\sqrt{\frac{1}{\epsilon} \cdot \frac{F(X^{(0)}) - F^{\infty}}{\mathsf{K}+1}}$$

for any $\mathsf{K} \ge 0$;

(f) if $\zeta > \xi > 0$, then $\mathrm{grad}\, F(X^{(\mathsf{k})}) \to \mathbf{0}$ and $\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})}) \to \mathbf{0}$ as $\mathsf{k} \to \infty$;

Proof. See Section 4.12.7. □

In spite of limited local communication, the AMM–PGO$^{\#}$ method has provable convergence as long as each node $\alpha \in \mathcal{A}$ can communicate with its neighbors $\beta \in \mathcal{N}^{\alpha}$. Thus, the AMM–PGO$^{\#}$ method eliminates the bottleneck of communication for distributed PGO without a master node in contrast to the AMM–PGO* method and [**59**]. In addition, note that the AMM–PGO$^{\#}$ method results in $F(X^{(\mathsf{k}+1)}) \le G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \le \overline{F}^{(\mathsf{k})}$ instead of $F(X^{(\mathsf{k}+1)}) \le G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \le F(X^{(\mathsf{k})})$ in [**1**]. Recalling $F(X^{(\mathsf{k})}) \le \overline{F}^{(\mathsf{k})}$, we conclude that the AMM–PGO$^{\#}$ method prevents unnecessary adaptive restarts and mitigates the impacts that $G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})})$ is an upper-bound of $F(X^{(\mathsf{k}+1)})$, which is expected to make better use of Nesterov's method for acceleration and have faster convergence than [**1**].

4.10. Experiments

In this section, we evaluate the performance of our MM methods (MM–PGO, AMM–PGO* and AMM–PGO$^{\#}$) for distributed PGO on the simulated Cube datasets and a number of 2D and 3D SLAM benchmark datasets [**7**]. In terms of MM–PGO, AMM–PGO* and AMM–PGO$^{\#}$, η, ξ, ζ, ψ and ϕ in Algorithms 9, 11 and 13 are 5×10^{-4}, 1×10^{-10}, 1.5×10^{-10}, 1×10^{-10} and 1×10^{-6}, respectively, for all the experiments. In addition,

MM–PGO, AMM–PGO* and AMM–PGO$^\#$ can take at most one iteration when solving Eqs. (4.59) and (4.76) to improve the estimates. All the experiments have been performed on a laptop with an Intel Xeon(R) CPU E3-1535M v6 and 64GB of RAM running Ubuntu 18.04.

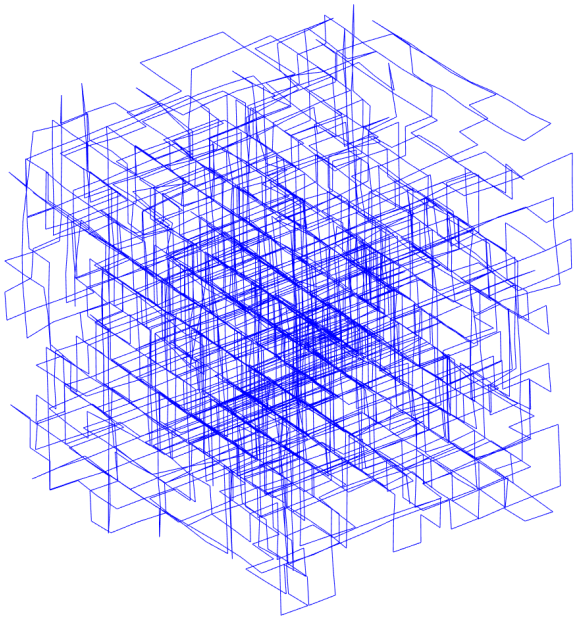


Figure 4.2. A Cube dataset has $12 \times 12 \times 12$ grids of side length of 1 m, 3600 poses, probability of loop closure of 0.1, an translational RSME of $\sigma_t = 0.02$ m and an angular RSME of $\sigma_R = 0.02\pi$ rad.

4.10.1. Cube Datasets

In this section, we test and evaluate our MM methods for distributed PGO on 20 simulated Cube datasets (see Fig. 4.2) with 5, 10 and 50 robots.

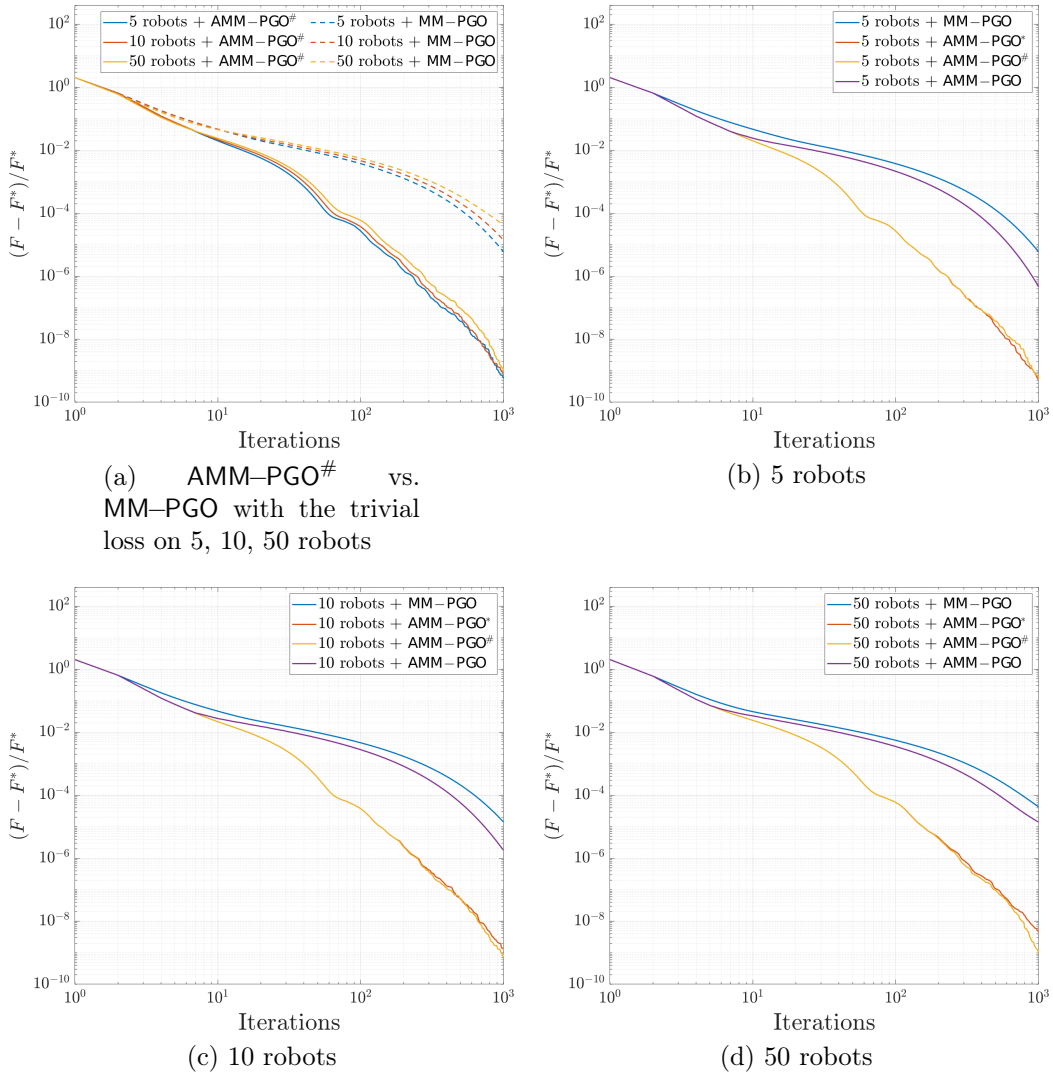
In the experiment, a simulated Cube dataset has $12 \times 12 \times 12$ cube grids with 1 m side length, a path of 3600 poses along the rectilinear edge of the cube grid, odometric measurements between all the pairs of sequential poses, and loop-closure measurements between nearby but non-sequential poses that are randomly available with a probability

of 0.1. We generate the odometric and loop-closure measurements according to the noise models in [**7**] with an expected translational RMSE of $\sigma_t = 0.02$ m and an expected angular RMSE of $\sigma_R = 0.02\pi$ rad. The centralized chordal initialization [**8**] is implemented such that distributed PGO with different number of robots have the same initial estimate. The maximum number of iterations is 1000.

We evaluate the convergence of MM–PGO, AMM–PGO* and AMM–PGO$^\#$ in terms of the relative suboptimality gap and Riemannian gradient norm. For reference, we also make comparisons against AMM–PGO [**1**]. Note that AMM–PGO is the original accelerated MM method for distributed PGO whose adaptive restart scheme is conservative and might prohibit Nesterov's acceleration.

Relative Suboptimality Gap. We implement the certifiably-correct SE−Sync [**7**] to get the globally optimal objective value F^* for distributed PGO with the trivial loss kernel (Example 4.1), making it possible to compute the relative suboptimality gap $(F - F^*)/F^*$ where F is the objective value for each iteration. The results are in Fig. 4.3.

Riemannian Gradient Norm. We also compute the Riemannian gradient norm for distributed PGO with the trivial (Example 4.1) and nontrivial—Huber (Example 4.2) and Welsch (Example 4.3)—losses kernels for evaluation. Note that it is difficult to find the globally optimal solution to distributed PGO if nontrivial loss kernels are used. The results are in Figs. 4.4 to 4.6.



(a) AMM–PGO$^{\#}$ vs. MM–PGO with the trivial loss on 5, 10, 50 robots

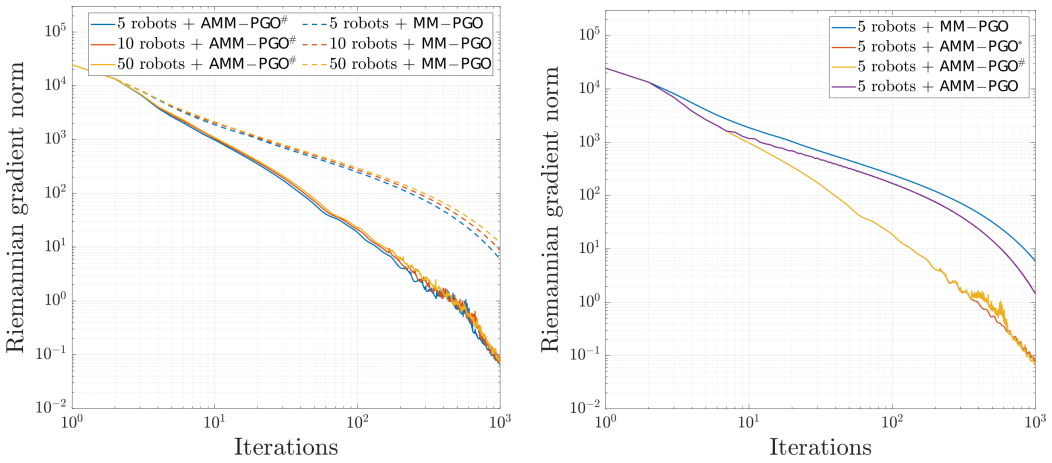
(b) 5 robots

(c) 10 robots

(d) 50 robots

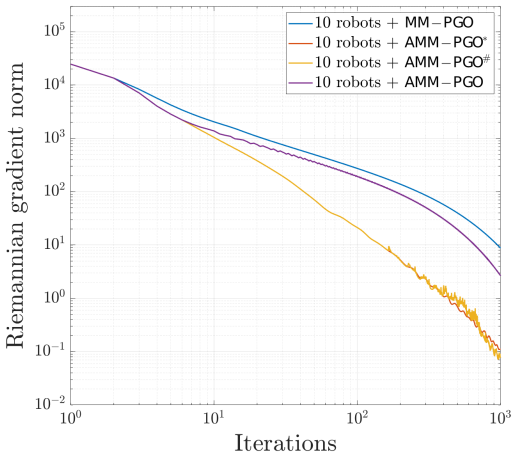
Figure 4.3. The relative suboptimality gaps of the MM–PGO, AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO [**1**] methods for distributed PGO with the trivial loss kernel on 5, 10 and 50 robots. The results are averaged over 20 Monte Carlo runs.

In Figs. 4.3 to 4.6, it can be seen that MM–PGO, AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO have a faster convergence if the number of robots (nodes) decreases. This is expected since $G(X|X^{(k)})$ and $H(X|X^{(k)})$ in Eqs. (4.29) and (4.35) result in tighter approximations for distributed PGO with fewer robots (nodes). In addition, Figs. 4.4 to 4.6 suggest that the convergence rate of MM–PGO, AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO

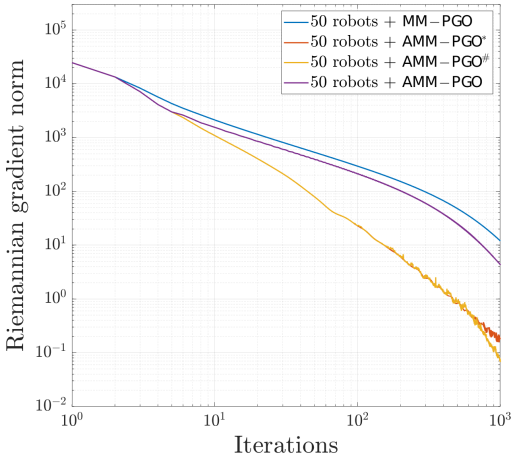


(a) AMM–PGO$^{\#}$ vs. MM–PGO with the trivial loss on 5, 10, 50 robots

(b) 5 robots



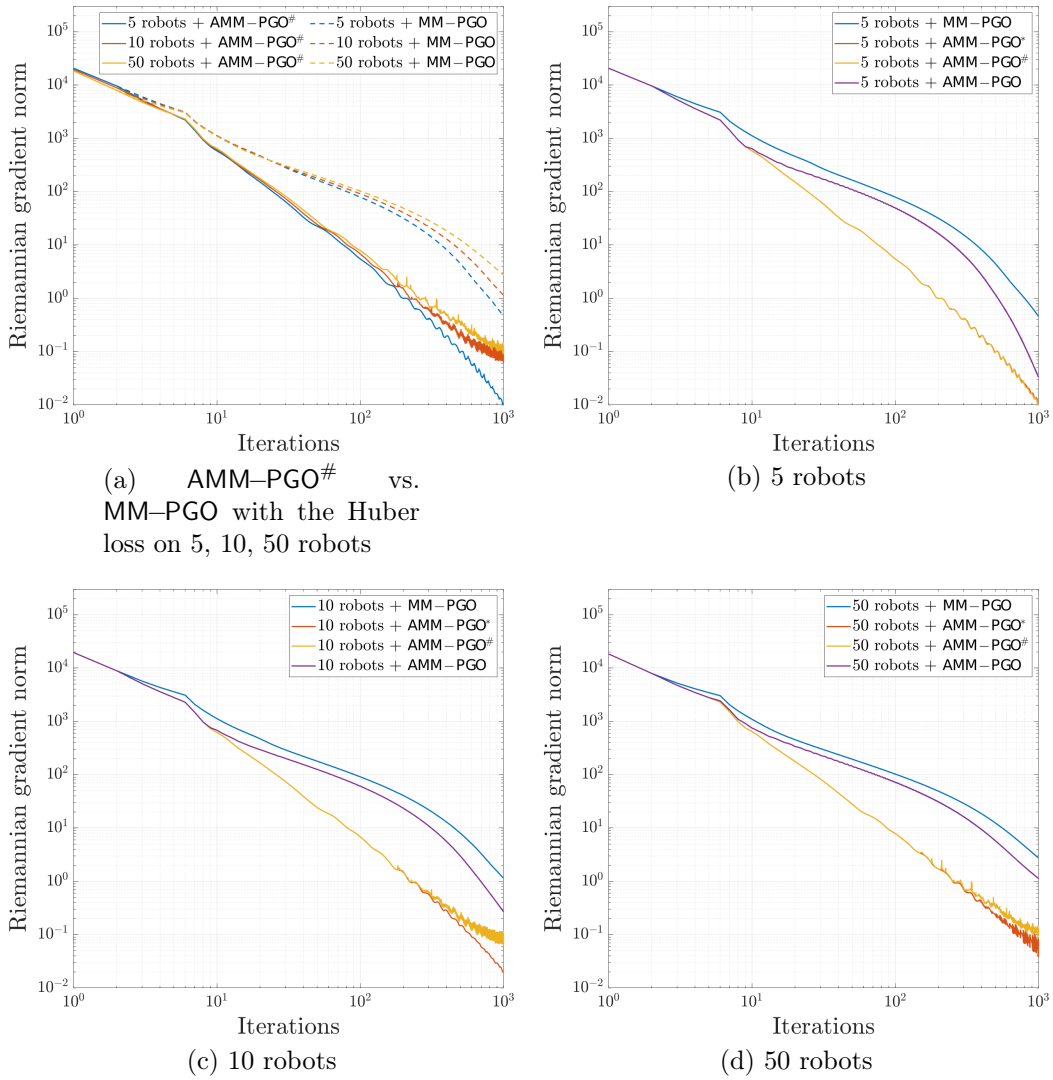
(c) 10 robots



(d) 50 robots

Figure 4.4. The Riemannian gradient norms of the MM–PGO, AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO [**1**] methods for distributed PGO with the trivial loss kernel on 5, 10 and 50 robots. The results are averaged over 20 Monte Carlo runs.

also relies on the type of loss kernels. Nevertheless, AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO accelerated by Nesterov's method outperform the unaccelerated MM–PGO method by a large margin for any number of robots and any types of loss kernels, which



(a) AMM–PGO$^{\#}$ vs. MM–PGO with the Huber loss on 5, 10, 50 robots

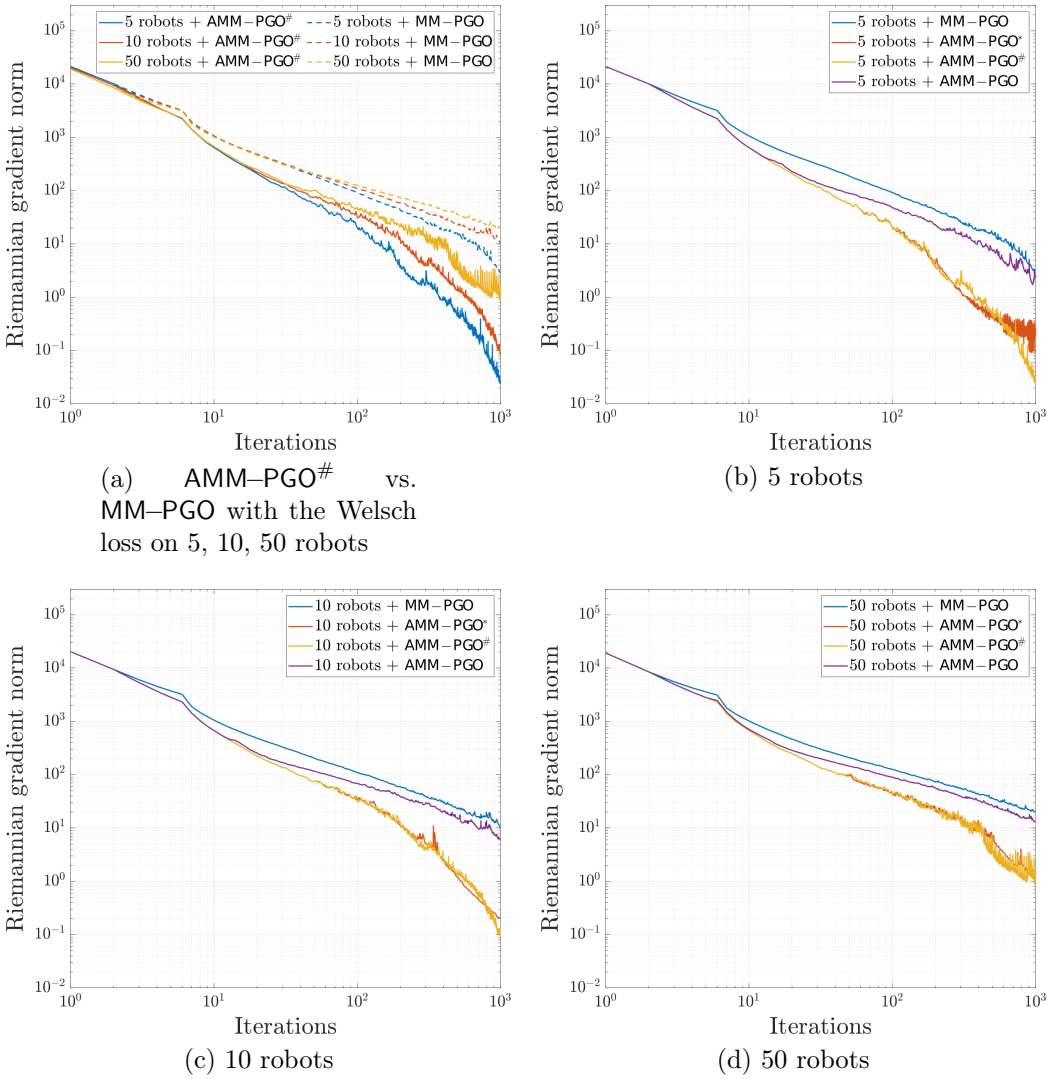
(b) 5 robots

(c) 10 robots

(d) 50 robots

Figure 4.5. The Riemannian gradient norms of the MM–PGO, AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO [**1**] methods for distributed PGO with the Huber loss kernel on 5, 10 and 50 robots. The results are averaged over 20 Monte Carlo runs.

means that Nesterov's method improves the convergence of distributed PGO. In particular, Figs. 4.3(a), 4.4(a), 4.5(a) and 4.6(a) indicate that AMM–PGO$^{\#}$ with 50 robot still converges faster than MM–PGO with 5 robots despite that the later has a much smaller



(a) AMM–PGO$^{\#}$ vs. MM–PGO with the Welsch loss on 5, 10, 50 robots

(b) 5 robots

(c) 10 robots

(d) 50 robots

Figure 4.6. The Riemannian gradient norms of the MM–PGO, AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO [**1**] methods for distributed PGO with the Welsch loss kernel on 5, 10 and 50 robots. The results are averaged over 20 Monte Carlo runs.

number of robots. Therefore, we conclude that the implementation of Nesterov's method accelerate the convergence of distributed PGO.

Furthermore, we emphasize the convergence comparisons of AMM–PGO*, AMM–PGO$^{\#}$ and AMM–PGO, which are all accelerated with Nesterov's method while differing from

each other by the adaptive restart schemes—AMM–PGO* has an additional master node to aggregate information from all the robots (nodes), whereas AMM–PGO$^\#$ and AMM–PGO are restricted to one inter-node communication round per iteration among neighboring robots (nodes). Notwithstanding limited local communication, as is shown in Figs. 4.3 to 4.6, AMM–PGO$^\#$ has a convergence rate comparable to that of AMM–PGO* using a master node while being significantly faster than AMM–PGO. In particular, AMM–PGO$^\#$ reduces adaptive restarts by 80% to 95% compared to AMM–PGO on the Cube datasets, and thus, is expected to make better use of Nesterov's acceleration. Since AMM–PGO$^\#$ and AMM–PGO differ in the adaptive restart schemes, we attribute the faster convergence of AMM–PGO$^\#$ to its redesigned adaptive restart scheme. These results suggest that AMM–PGO$^\#$ is advantageous over other methods for very large-scale distributed PGO where computational and communicational efficiency are equally important.

4.10.2. Benchmark Datasets

In this section, we evaluate our MM methods (MM–PGO, AMM–PGO* and AMM–PGO$^\#$) for distributed PGO on a number of 2D and 3D SLAM benchmark datasets (see Table 4.1) [**7**]. We use the trivial loss kernel and assume that there are no outliers such that the globally optimal solution can be exactly computed with SE−Sync [**7**]. For each dataset, we also make comparisons against SE−Sync [**7**], distributed Gauss-Seidel (DGS) [**5**] and the Riemannian block coordinate descent (RBCD) [**6**] method, all of which are the state-of-the-art algorithms for centralized and distributed PGO. The SE−Sync and DGS methods use the recommended settings in [**5**, **7**]. We implement two Nesterov's accelerated variants of RBCD [**6**], i.e., one with greedy selection rule and adaptive restart

(RBCD++*) and the other with uniform selection rule and fixed restart (RBCD++$^\#$)[2]. As mentioned before, AMM–PGO* and AMM–PGO$^\#$ can take at most one iteration when updating $X^{\alpha(k+1)}$ using Eqs. (4.59) and (4.76), which is similar to RBCD++* and RBCD++$^\#$. An overview of the aforementioned methods is given in Table 4.2.

Number of Iterations. First, we examine the convergence of MM–PGO, AMM–PGO*, AMM–PGO$^\#$, DGS [**5**], RBCD++* [**6**] and RBCD++$^\#$ [**6**] w.r.t. the number of iterations. The distributed PGO has 10 robots and all the methods are initialized with the distributed Nesterov's accelerated chordal initialization [**1**].

The reconstruction results using AMM–PGO$^\#$ are shown in Figs. 4.7 and 4.8 and the objective values of each method with 100, 250 and 1000 iterations are reported in Tables 4.3 and 4.4. For almost all the benchmark datasets, AMM–PGO* and AMM–PGO$^\#$ outperform the other methods (MM–PGO, DGS, RBCD++* and RBCD++$^\#$). While RBCD++* and RBCD++$^\#$ have similar performances in four comparatively simple datasets—CSAIL, sphere, torus and grid—we remark that AMM–PGO* and AMM–PGO$^\#$ achieve much better results in the other more challenging datasets in particular if there are no more than 250 iterations. As discussed later, AMM–PGO* and AMM–PGO$^\#$ have faster convergence to more accurate estimates without any extra computation and communication in contrast to RBCD++* and RBCD++$^\#$. Last but not the least, Tables 4.3 and 4.4 demonstrate that the accelerated AMM–PGO* and AMM–PGO$^\#$ converge significantly faster than the unaccelerated MM–PGO, which further validates the usefulness of Nesterov's method.

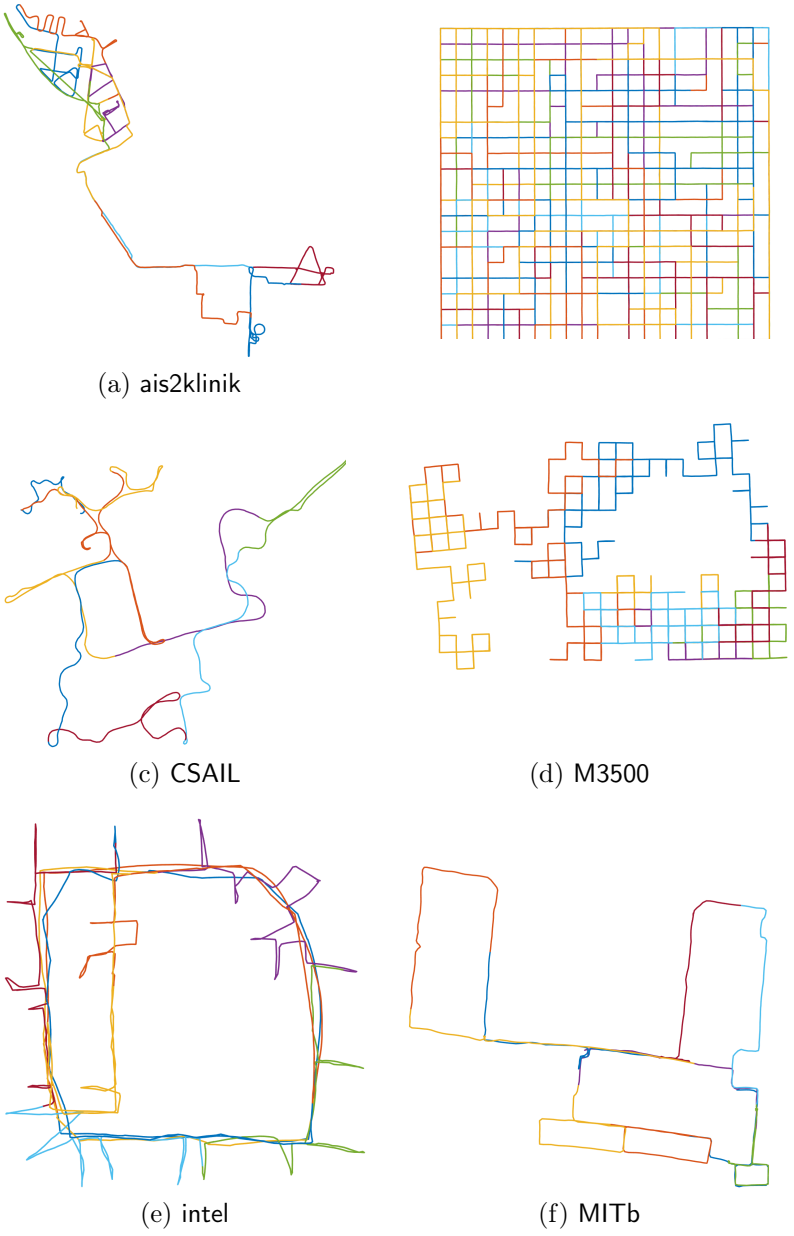
[2]In the experiments, we run RBCD++$^\#$ [**6**] with fixed restart frequencies of 30, 50 and 100 iterations for each dataset and select the one with the best performance.

Table 4.1. 2D and 3D SLAM benchmark datasets.

Dataset	2D/3D	# Poses	# Measurements	Simulated Dataset
ais2klinik	2D	15115	16727	No
city	2D	10000	20687	Yes
CSAIL	2D	1045	1172	No
M3500	2D	3500	5453	Yes
intel	2D	1728	2512	No
MITb	2D	808	827	No
sphere	3D	2500	4949	Yes
torus	3D	5000	9048	Yes
grid	3D	8000	22236	Yes
garage	3D	1661	6275	No
cubicle	3D	5750	16869	No
rim	3D	10195	29743	No

Table 4.2. An overview of the state-of-the-art algorithms for distributed and centralized PGO. Note that AMM–PGO* and RBCD++* require a master node for distributed PGO. In addition, AMM–PGO$^\#$ is the only accelerated method for distributed PGO that has provable convergence without a master node.

Method	Distributed	Accelerated	Masterless	Converged
SE−Sync [**7**]	No	N/A	N/A	Yes
DGS [**5**]	Yes	No	Yes	No
RBCD++* [**6**]	Yes	Yes	No	Yes
RBCD++$^\#$ [**6**]	Yes	Yes	Yes	No
MM–PGO	Yes	No	Yes	Yes
AMM–PGO*	Yes	Yes	No	Yes
AMM–PGO$^\#$	Yes	Yes	Yes	Yes



(a) ais2klinik

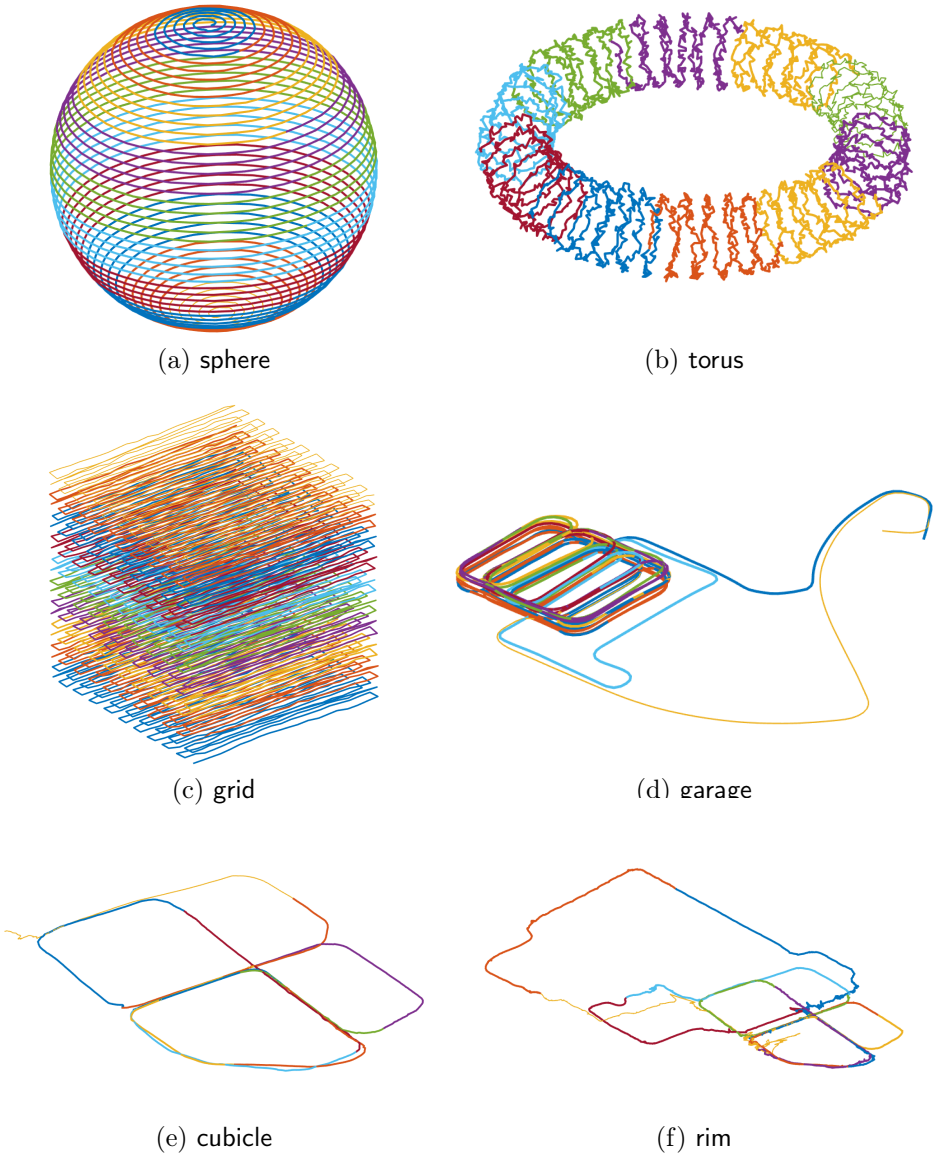
(c) CSAIL

(d) M3500

(e) intel

(f) MITb

Figure 4.7. AMM–PGO$^{\#}$ results on the 2D SLAM benchmark datasets where the different colors denote the odometries of different robots. The distributed PGO has 10 robots and is initialized with the distributed Nesterov's accelerated chordal initialization [**1**]. The number of iterations is 1000.



(a) sphere (b) torus

(c) grid (d) garage

(e) cubicle (f) rim

Figure 4.8. AMM–PGO$^{\#}$ results on the 3D SLAM benchmark datasets where the different colors denote the odometries of different robots. The distributed PGO has 10 robots and is initialized with the distributed Nesterov's accelerated chordal initialization [**1**]. The number of iterations is 1000.

Table 4.3. Results of distributed PGO on the 2D SLAM Benchmark datasets (see Table 4.1). The distributed PGO has 10 robots and is initialized with the distributed Nesterov's accelerated chordal initialization [**1**]. We report the objective values of each method with 100, 250 and 1000 iterations. $F^{(\mathsf{k})}$ and F^* are the objective value at iteration k and globally optimal objective value, respectively. The best results are colored in red and the second best in blue if no methods tie for the best.

Dataset	$F^{(0)}$	F^*	k	$F^{(\mathsf{k})}$					
				Methods w/ Master Node		Methods w/o Master Node			
				AMM–PGO*	RBCD++* [6]	MM–PGO	AMM–PGO#	DGS [5]	RBCD++# [6]
2D SLAM Benchmark Datasets									
ais2klinik	3.8375×10^2	1.8850×10^2	100	2.0372×10^2	2.1079×10^2	2.1914×10^2	2.0371×10^2	8.4701×10^2	2.1715×10^2
			250	1.9447×10^2	2.0077×10^2	2.1371×10^2	1.9446×10^2	9.1623×10^1	2.1084×10^2
			1000	1.8973×10^2	1.9074×10^2	2.0585×10^2	1.8936×10^2	3.8968×10^2	2.0253×10^2
city	7.0404×10^2	6.3862×10^2	100	6.4327×10^2	6.5138×10^2	6.5061×10^2	6.4327×10^2	7.7745×10^2	6.5396×10^2
			250	6.3899×10^2	6.4732×10^2	6.4850×10^2	6.3899×10^2	7.0063×10^2	6.5122×10^2
			1000	6.3862×10^2	6.3935×10^2	6.4461×10^2	6.3863×10^2	6.5583×10^2	6.4768×10^2
CSAIL	3.1719×10^1	3.1704×10^1	100	3.1704×10^1	3.1704×10^1	3.1706×10^1	3.1704×10^1	3.2479×10^1	3.1705×10^1
			250	3.1704×10^1	3.1704×10^1	3.1706×10^1	3.1704×10^1	3.1792×10^1	3.1704×10^1
			1000	3.1704×10^1	3.1704×10^1	3.1705×10^1	3.1704×10^1	3.1712×10^1	3.1704×10^1
M3500	2.2311×10^2	1.9386×10^2	100	1.9446×10^2	1.9511×10^2	1.9560×10^2	1.9447×10^2	1.9557×10^2	1.9551×10^2
			250	1.9414×10^2	1.9443×10^2	1.9516×10^2	1.9414×10^2	1.9445×10^2	1.9511×10^2
			1000	1.9388×10^2	1.9392×10^2	1.9461×10^2	1.9388×10^2	1.9415×10^2	1.9455×10^2
intel	5.3269×10^1	5.2348×10^1	100	5.2397×10^1	5.2496×10^1	5.2517×10^1	5.2397×10^1	5.2541×10^1	5.2526×10^1
			250	5.2352×10^1	5.2415×10^1	5.2483×10^1	5.2351×10^1	5.2441×10^1	5.2489×10^1
			1000	5.2348×10^1	5.2349×10^1	5.2421×10^1	5.2348×10^1	5.2381×10^1	5.2425×10^1
MITb	8.8430×10^1	6.1154×10^1	100	6.1331×10^1	6.1518×10^1	6.3657×10^1	6.1330×10^1	9.5460×10^1	6.1997×10^1
			250	6.1157×10^1	6.1187×10^1	6.2335×10^1	6.1165×10^1	7.8273×10^1	6.1599×10^1
			1000	6.1154×10^1	6.1154×10^1	6.1454×10^1	6.1154×10^1	7.2450×10^1	6.1209×10^1

Table 4.4. Results of distributed PGO on the 3D SLAM Benchmark datasets (see Table 4.1). The distributed PGO has 10 robots and is initialized with the distributed Nesterov's accelerated chordal initialization [**1**]. We report the objective values of each method with 100, 250 and 1000 iterations. $F^{(\mathsf{k})}$ and F^* are the objective value at iteration k and globally optimal objective value, respectively. The best results are colored in red and the second best in blue if no methods tie for the best.

Dataset	$F^{(0)}$	F^*	k	$F^{(\mathsf{k})}$					
				Methods w/ Master Node		Methods w/o Master Node			
				AMM–PGO*	RBCD++* [**6**]	MM–PGO	AMM–PGO$^\#$	DGS [**5**]	RBCD++$^\#$ [**6**]
3D SLAM Benchmark Datasets									
sphere	1.9704×10^3	1.6870×10^3	100	1.6870×10^3	1.6870×10^3	1.6901×10^3	1.6870×10^3	1.6875×10^3	1.6870×10^3
			250	1.6870×10^3	1.6870×10^3	1.6874×10^3	1.6870×10^3	1.6872×10^3	1.6870×10^3
			1000	1.6870×10^3	1.6870×10^3	1.6870×10^3	1.6870×10^3	1.6872×10^3	1.6870×10^3
torus	2.4654×10^4	2.4227×10^4	100	2.4227×10^4	2.4227×10^4	2.4234×10^4	2.4227×10^4	2.4248×10^4	2.4227×10^4
			250	2.4227×10^4	2.4227×10^4	2.4227×10^4	2.4227×10^4	2.4243×10^4	2.4227×10^4
			1000	2.4227×10^4	2.4227×10^4	2.4227×10^4	2.4227×10^4	2.4236×10^4	2.4227×10^4
grid	2.8218×10^5	8.4319×10^4	100	8.4323×10^4	8.4320×10^4	1.0830×10^5	8.4399×10^4	1.4847×10^5	8.4920×10^4
			250	8.4319×10^4	8.4319×10^4	8.6054×10^4	8.4321×10^4	1.4066×10^5	8.4319×10^4
			1000	8.4319×10^4	8.4319×10^4	8.4319×10^4	8.4319×10^4	1.4654×10^5	8.4319×10^4
garage	1.5470×10^0	1.2625×10^0	100	1.3105×10^0	1.3282×10^0	1.3396×10^0	1.3105×10^0	1.3170×10^0	1.3364×10^0
			250	1.2872×10^0	1.3094×10^0	1.3288×10^0	1.2872×10^0	1.2867×10^0	1.3276×10^0
			1000	1.2636×10^0	1.2681×10^0	1.3145×10^0	1.2636×10^0	1.2722×10^0	1.3124×10^0
cubicle	8.3514×10^2	7.1713×10^2	100	7.1812×10^2	7.2048×10^2	7.2300×10^2	7.1812×10^2	7.3185×10^2	7.2210×10^2
			250	7.1714×10^2	7.1794×10^2	7.2082×10^2	7.1715×10^2	7.2308×10^2	7.2081×10^2
			1000	7.1713×10^2	7.1713×10^2	7.2082×10^2	7.1713×10^2	7.2044×10^2	7.1845×10^2
rim	8.1406×10^4	5.4609×10^3	100	5.5044×10^3	5.7184×10^3	5.8138×10^3	5.5044×10^3	6.1840×10^3	5.7810×10^3
			250	5.4648×10^3	5.5050×10^3	5.7197×10^3	5.4648×10^3	6.1184×10^3	5.7195×10^3
			1000	5.4609×10^3	5.4617×10^3	5.5509×10^3	5.4609×10^3	6.0258×10^3	5.5373×10^3

We also compute the performance profiles [**136**] based on the number of iterations. Given a tolerance $\Delta \in (0,\, 1]$, the objective value threshold $F_\Delta(p)$ for PGO problem p is defined to be

$$F_\Delta(p) = F^* + \Delta \cdot \big(F^{(0)} - F^*\big), \tag{4.93}$$

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

$$I_\Delta(p) \triangleq \min_{\mathsf{k}} \big\{\mathsf{k} \geq 0 | F^{(\mathsf{k})} \leq F_\Delta(p)\big\},$$

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

$$\underset{\text{at iteration } \mathsf{k}}{\text{percentage of problems solved}} \triangleq \frac{\big|\{p \in \mathcal{P} | I_\Delta(p) \leq \mathsf{k}\}\big|}{|\mathcal{P}|}.$$

The performance profiles based on the number of iterations over a variety of 2D and 3D SLAM benchmark datasets (see Table 4.1) are shown in Fig. 4.9. The tolerances evaluated are $\Delta = 1 \times 10^{-2}$, 5×10^{-3}, 1×10^{-3} and 1×10^{-4}. We report the performance of MM–PGO, AMM–PGO*, AMM–PGO$^\#$, DGS [**5**], RBCD++* [**6**] and RBCD++$^\#$ [**6**] for distributed PGO with 10 robots (nodes). As expected, AMM–PGO* and AMM–PGO* dominates the other methods (MM–PGO, DGS, RBCD++* and RBCD++$^\#$) in terms of the convergence for all the tolerances Δ, which means that AMM–PGO* and AMM–PGO* are better choices for distributed PGO.

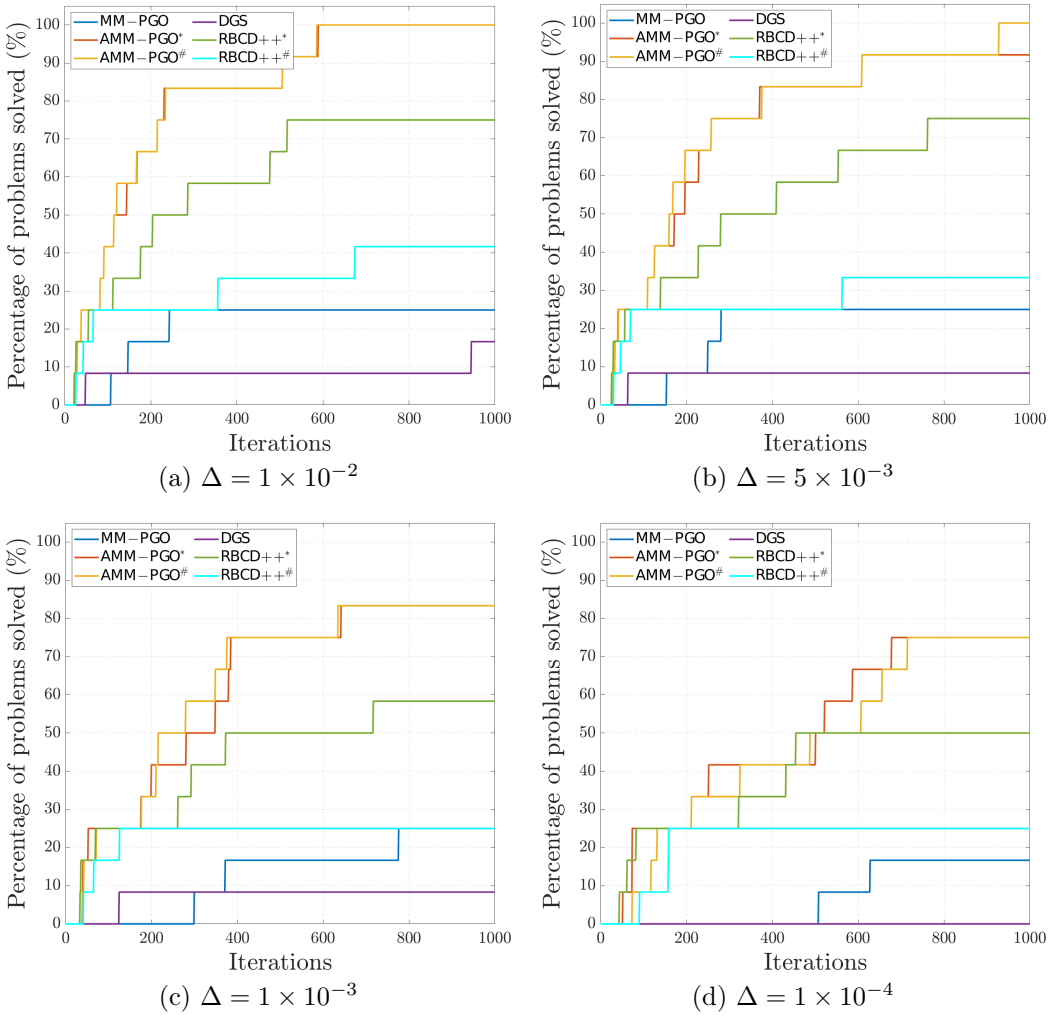


Figure 4.9. Performance profiles for MM–PGO, AMM–PGO*, AMM–PGO$^\#$, DGS [**5**], RBCD++* [**6**] and RBCD++* [**6**] over a variety of 2D and 3D SLAM Benchmark datasets (see Table 4.1). The performance is based on the number of iterations k and the evaluation tolerances are $\Delta = 1 \times 10^{-2}$, 5×10^{-3}, 1×10^{-3} and 1×10^{-4}. The distributed PGO has 10 robots (nodes) and is initialized with the distributed Nesterov's accelerated chordal initialization [**1**]. Note that AMM–PGO* and RBCD++* [**6**] require a master node, whereas MM–PGO, AMM–PGO$^\#$, DGS [**5**] and RBCD++$^\#$ [**6**] do not.

In Tables 4.3 and 4.4 and Fig. 4.9, we emphasize that AMM–PGO$^\#$ requiring no master node achieves comparable performance to that of AMM–PGO* using a master node, and

is a lot better than all the other methods with a master node (RBCD++*) and without (MM–PGO, DGS and RBCD++$^\#$). Even though RBCD++* and RBCD++$^\#$ are similarly accelerated with Nesterov's method, we remark that RBCD++$^\#$ without a master node suffers a great performance drop compared to RBCD++*, and more importantly, RBCD++$^\#$ has no convergence guarantees to first-order critical points. These results reverify that AMM–PGO$^\#$ is more suitable for very large-scale distributed PGO with limited local communication.

Note that all of MM–PGO, AMM–PGO*, AMM–PGO$^\#$, DGS [**5**], RBCD++* [**6**] and RBCD++$^\#$ [**6**] have to exchange poses of inter-node measurements with the neighbors, and thus, need almost the same amount of communication per iteration. However, Fig. 4.9 indicates that AMM–PGO* and AMM–PGO$^\#$ have much faster convergence in terms of the number of iterations, which also means less communication for the same level of accuracy. In addition, RBCD++* and RBCD++$^\#$ have to keep part of the nodes in idle during optimization and rely on red-black coloring for block aggregation and random sampling for block selection, which induce additional computation and communication. In contrast, neither AMM–PGO* nor AMM–PGO$^\#$ has any extra practical restrictions except those in Assumptions 4.1 to 4.4.

Optimization Time. In addition, we evaluate the speedup of AMM–PGO* and AMM–PGO$^\#$ with different numbers of robots (nodes) against the state-of-the-art centralized algorithm SE−Sync [**7**]. To improve the optimization time efficiency of AMM–PGO* and AMM–PGO$^\#$, $X^{\alpha(\mathrm{k}+1)}$ in Eqs. (4.59) and (4.76) uses the same rotation as $X^{\alpha(\mathrm{k}+\frac{1}{2})}$ and only updates the translation. Since the number of robots varies in the experiments, the centralized chordal initialization [**8**] is adopted for all the methods.

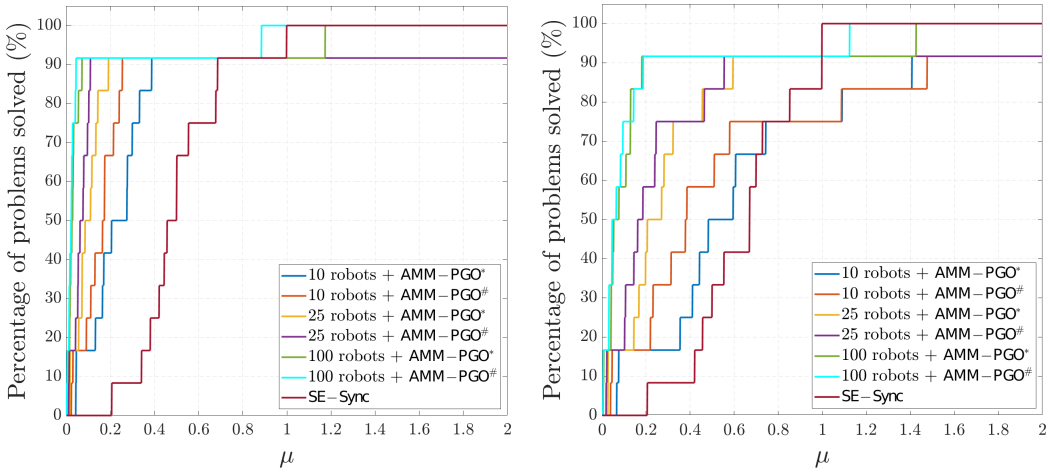
Similar to the number of iterations, we also use the performance profiles to evaluate AMM–PGO* and AMM–PGO$^\#$ in terms of the optimization time. Recall from Eq. (4.93) the objective value threshold $F_\Delta(p)$ where p is the PGO problem and $\Delta \in (0, 1]$ is the tolerance. Since the average optimization time per node is directly related with the speedup, we measure the efficiency of a distributed PGO method with N nodes by computing the average optimization time $T_\Delta(p, N)$ that each node takes to reduce the objective value to $F_\Delta(p)$:

$$T_\Delta(p, N) = \frac{T_\Delta(p)}{N},$$

where $T_\Delta(p)$ denotes the total optimization time of all the N nodes. We remark that the centralized optimization method has $N = 1$ node and $T_\Delta(p, N) = T_\Delta(p)$. Let $T_{\mathsf{SE\text{–}Sync}}$ denote the optimization time that $\mathsf{SE-Sync}$ needs to find the globally optimal solution. The performance profiles assume a distributed PGO method solves problem p for some $\mu \in [0, +\infty)$ if $T_\Delta(p, N) \leq \mu \cdot T_{\mathsf{SE\text{–}Sync}}$. Note that μ is the scaled average optimization time per node and $\mathsf{SE-Sync}$ solves problem p globally at $\mu = 1$. Then, as a result of **[136]**, the performance profiles evaluate the speedup of distributed PGO methods for a given optimization problem set $\mathcal{P}$ using the percentage of problems solved w.r.t. the scaled average optimization time per node $\mu \in [0, +\infty)$:

$$\underset{\text{solved at } \mu}{\text{percentage of problems}} \triangleq \frac{\left|\{p \in \mathcal{P} | T_\Delta(p, N) \leq \mu \cdot T_{\mathsf{SE\text{–}Sync}}\}\right|}{|\mathcal{P}|}.$$

Fig. 4.10 shows the performance profiles based on the scaled average optimization time per node. The tolerances evaluated are $\Delta = 1 \times 10^{-2}$, 1×10^{-3}, 1×10^{-4} and 1×10^{-5}. We report the performance of AMM–PGO* and AMM–PGO$^\#$ with 10, 25



(a) $\Delta = 1 \times 10^{-2}$ (b) $\Delta = 1 \times 10^{-3}$

(c) $\Delta = 1 \times 10^{-4}$ (d) $\Delta = 1 \times 10^{-5}$

Figure 4.10. Performance profiles for AMM–PGO*, AMM–PGO$^\#$ and SE−Sync [7] over a variety of 2D and 3D SLAM Benchmark datasets (see Table 4.1). The performance is based on the scaled average optimization time per node $\mu \in [0, +\infty)$ and the evaluation tolerances are $\Delta = 1 \times 10^{-2}$, 1×10^{-3}, 1×10^{-4} and 1×10^{-5}. The distributed PGO has 10, 25 and 100 robots (nodes) and is initialized with the classic chordal initialization [8]. Note that SE−Sync [7] solves all the PGO problems globally at $\mu = 1$.

and 100 robots (nodes). For reference, we also evaluate the performance profile of the centralized PGO baseline SE−Sync [7]. As the results demonstrate, AMM–PGO* and

AMM–PGO$^{\#}$ are significantly faster than SE$-$Sync [**7**] in most cases for modest accuracies of $\Delta = 1\times10^{-2}$ and $\Delta = 1\times10^{-3}$, for which the only challenging case is the CSAIL dataset, whose chordal initialization is already very close to the globally optimal solution. Even though the performance of AMM–PGO* and AMM–PGO$^{\#}$ declines for smaller tolerances of $\Delta = 1\times10^{-4}$ and $\Delta = 1\times10^{-5}$, AMM–PGO* and AMM–PGO$^{\#}$ with 100 robots (nodes) still achieve a $2.5 \sim 20$x speedup over SE$-$Sync for more than 70% of the benchmark datasets. Furthermore, in terms of the average optimization time per node, the computational efficiency of AMM–PGO* and AMM–PGO$^{\#}$ improves as the number of robots (nodes) increases, which indicates the possibility of using accelerated MM methods as fast parallel backends for real-time SLAM.

In summary, AMM–PGO* and AMM–PGO$^{\#}$ not only achieve the state-of-the-art performance for distributed PGO but also enjoy significant multi-node speedup against the centralized baseline [**7**] for modest accuracy that is sufficient for practical use.

4.10.3. Robust Distributed PGO

In this section, we evaluate the robustness of AMM–PGO$^{\#}$ against the outlier inter-node loop closures. Similar to [**112**, **115**], we first use the distributed pairwise consistent measurement set maximization algorithm (PCM) [**9**] to reject spurious inter-node loop closures and then solve the resulting distributed PGO using AMM–PGO$^{\#}$ with the trivial (Example 4.1), Huber (Example 4.2) and Welsch (Example 4.3) loss kernels.

We implement AMM–PGO$^{\#}$ on the 2D intel and 3D garage datasets (see Table 4.1) with 10 robots (nodes). For each dataset, we add false inter-node loop closures with uniformly random rotation and translation errors in the range of [0, π] rad and [0, 5] m, respectively.

In addition, after the initial outlier rejection using the PCM algorithm [**9**], we initialize AMM–PGO$^{\#}$ with the distributed Nesterov's accelerated chordal initialization [**1**] for all the loss kernels.

The absolute trajectory errors (ATE) of AMM–PGO$^{\#}$ for different outlier thresholds of inter-node loop closures are reported in Fig. 4.11. The ATEs are computed against the outlier-free results of SE−Sync [**7**] and averaged over 10 Monte Carlo runs.

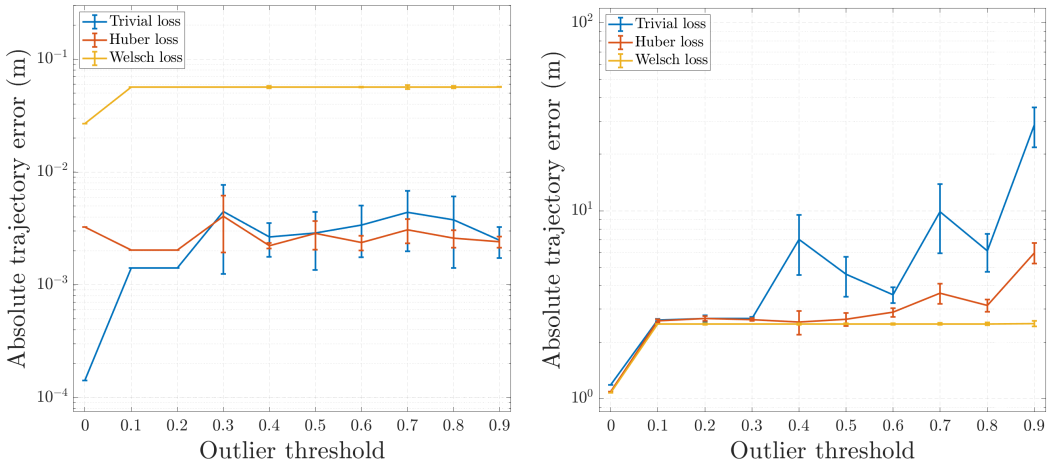


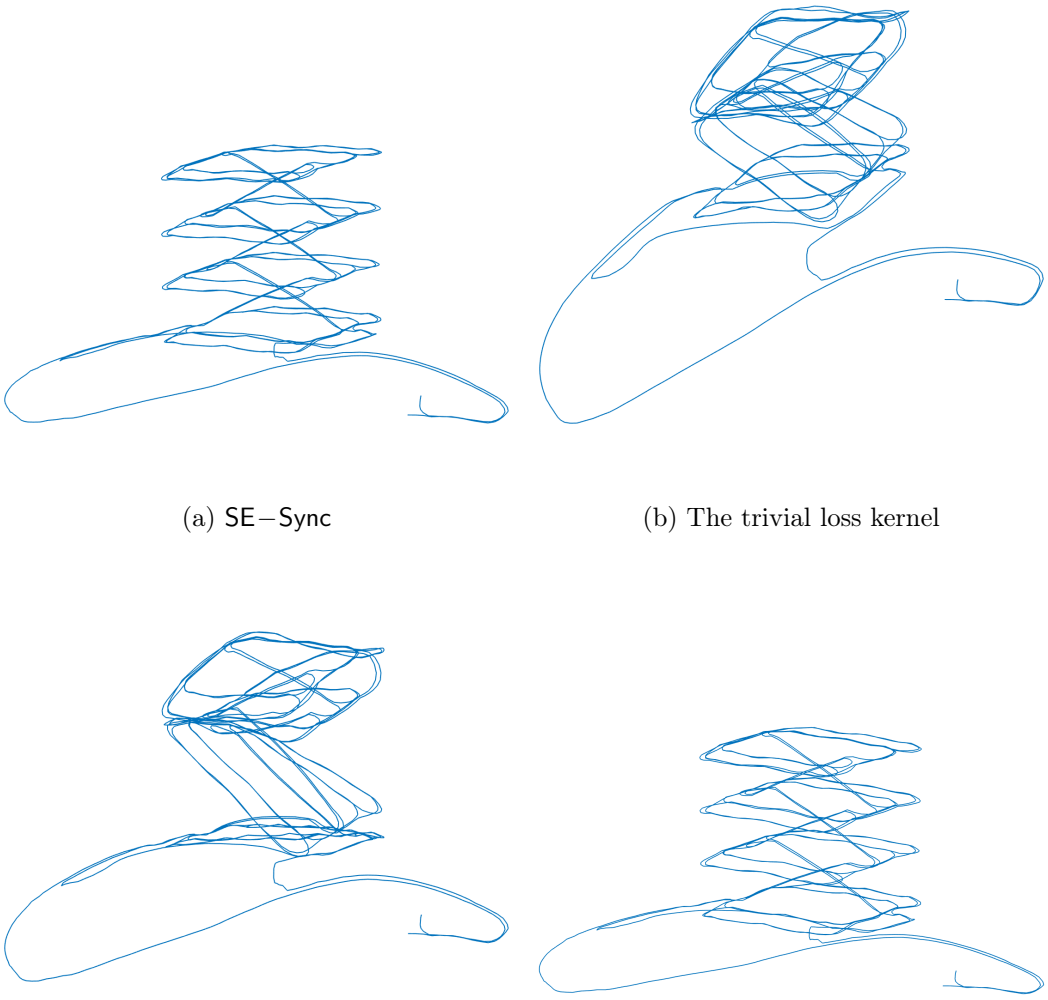
Figure 4.11. Absolute trajectory errors (ATE) of distributed PGO using AMM–PGO$^{\#}$ with the trivial, Huber and Welsch loss kernels on the 2D intel and 3D garage datasets. The outlier thresholds of inter-node loop closures are $0 \sim 0.9$. The ATEs are computed against the outlier-free results of SE−Sync [**7**] and are averaged over 10 Monte Carlo runs. The distributed PGO has 10 robots (nodes) and is initialized with the distributed Nesterov's accelerated chordal initialization [**1**]. The PCM algorithm [**9**] is used to initially reject spurious inter-robot loop closures.

In Fig. 4.11(a), PCM [**9**] rejects most of the outlier inter-node loop closure for the intel dataset and AMM–PGO$^{\#}$ solves the distributed PGO problems regardless of the loss kernel types and outlier thresholds. Note that AMM–PGO$^{\#}$ with the Welsch loss kernel has larger ATEs (avg. 0.057 m) against SE−Sync [**7**] than those with the trivial and

Huber loss kernels (avg. 0.003 m), and we argue that this is related to the loss kernel types. The ATEs are evaluated based on SE−Sync using the trivial loss kernel, which is in fact identical or similar to distributed PGO with the trivial and Huber loss kernels but different from that with the Welsch loss kernel. Therefore, the estimates from the trivial and Huber loss kernels are expected to be more close to those of SE−Sync, which result in smaller ATEs compared to the Welsch loss kernel if there are no outliers.

For the more challenging garage dataset, as is shown in Fig. 4.11(b), PCM fails for outlier thresholds over 0.4, and further, distributed PGO with the trivial and Huber loss kernels results in ATEs as large as 65 m. In contrast, distributed PGO with the Welsch loss kernel still successfully estimates the poses with an average ATE of 2.5 m despite the existence of outliers—note that the garage dataset has a trajectory over 7 km. For the garage dataset, a qualitative comparison of distributed PGO with different loss kernels is also presented in Fig. 4.12, where the Welsch loss kernel still has the best performance. The results are not surprising since the Welsch loss kernel is known to be more robust against outliers than the other two loss kernels [**129**].

The results above indicate that our MM methods can be applied to distributed PGO in the presence of outlier inter-node loop closures when combined with robust loss kernels like Welsch and other outlier rejection techniques like PCM [**9**]. In addition, we emphasize again that our MM methods have provable convergence to first-order critical points for a broad class of robust loss kernels, whereas the convergence guarantees of existing distributed PGO methods [**5**, **6**, **123**, **124**] are restricted to the trivial loss kernel.



(a) SE−Sync

(b) The trivial loss kernel

(c) The Huber loss kernel

(d) The Welsch loss kernel

Figure 4.12. A qualitative comparison of distributed PGO with the trivial, Huber and Welsch loss kernels for the garage dataset with spurious inter-node loop closures. The outlier-free result of SE−Sync [7] is shown in Fig. 4.12(a) for reference. The outlier threshold of inter-node loop closures is 0.6 and PCM [9] is used for initial outlier rejection.

4.11. Conclusion

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

Our MM methods for distributed PGO can be improved as follows. A more tractable and robust initialization technique is definitely beneficial to the accuracy and efficiency of distributed PGO. Even though our MM methods have reliable performances against outliers, a more complete theoretical analysis for robust distributed PGO is still necessary. In addition, our MM methods can be implemented as local solvers for distributed certifiably correct PGO **[6]** to handle poor or random initialization. Since all the nodes are now assumed to be synchronized, it is necessary and useful to extend our MM methods for asynchronous distributed PGO. Lastly, real multi-robot tests might make the results of our MM methods more convincing where not only the optimization time but also the communication overhead can be validated.

4.12. Proofs

4.12.1. Proof of Proposition 4.5.1

For any nodes α, $\beta \in \mathcal{A}$, it should be noted that

$$\frac{1}{2}\|X\|^2_{M_{ij}^{\alpha\beta}} = \frac{1}{2}\|X - X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}} + \left\langle X^{(\mathsf{k})} M_{ij}^{\alpha\beta}, X - X^{(\mathsf{k})} \right\rangle + \frac{1}{2}\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}} \tag{4.94}$$

always holds. Then, we will prove Proposition 4.5.1 considering cases of $\alpha = \beta$ and $\alpha \neq \beta$, respectively.

1) If $\alpha = \beta$, Eq. (4.16) indicates $\nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}) = X^{(\mathsf{k})} M_{ij}^{\alpha\beta}$. From Eqs. (4.16), (4.19) and (4.94), it is immediate to conclude that Eq. (4.18) holds for any X and $X^{(\mathsf{k})} \in \mathbb{R}^{d\times(d+1)n}$ as long as $\alpha = \beta$.

2) From Assumption 4.2(c), it is known that $\rho(s)$ is a concave function, which suggests

$$\rho(s') \leq \rho(s) + \nabla\rho(s) \cdot (s' - s).$$

If we let $s = \|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}$ and $s' = \|X\|^2_{M_{ij}^{\alpha\beta}}$, the equation above can be written as

$$\frac{1}{2}\rho\big(\|X\|^2_{M_{ij}^{\alpha\beta}}\big) \leq \frac{1}{2}\rho\big(\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big) + \frac{1}{2}\nabla\rho\big(\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big) \cdot \big(\|X\|^2_{M_{ij}^{\alpha\beta}} - \|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big). \tag{4.95}$$

From Eq. (4.94), it can be shown that

$$\frac{1}{2}\|X\|^2_{M_{ij}^{\alpha\beta}} - \frac{1}{2}\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}} = \frac{1}{2}\|X - X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}} + \left\langle X^{(\mathsf{k})} M_{ij}^{\alpha\beta}, X - X^{(\mathsf{k})} \right\rangle. \tag{4.96}$$

Then, applying Eq. (4.96) on the right-hand side of Eq. (4.95) results in

$$\frac{1}{2}\rho\big(\|X\|^2_{M_{ij}^{\alpha\beta}}\big) \leq \frac{1}{2}\nabla\rho\big(\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big) \cdot \|X - X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}} + \tag{4.97}$$

$$\nabla\rho\big(\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big)\cdot\big\langle X^{(\mathsf{k})}M_{ij}^{\alpha\beta},X-X^{(\mathsf{k})}\big\rangle+\frac{1}{2}\rho\big(\|X^{(\mathsf{k})}\|_{M_{ij}^{\alpha\beta}}\big).$$

From Eqs. (4.17) and (4.19), we obtain

$$F_{ij}^{\alpha\beta}(X^{(\mathsf{k})})=\frac{1}{2}\rho\big(\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big),$$

$$\nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})})=\nabla\rho\big(\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big)\cdot X^{(\mathsf{k})}M_{ij}^{\alpha\beta},$$

$$\omega_{ij}^{\alpha\beta(\mathsf{k})}=\nabla\rho\big(\|X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}\big),$$

with which Eq. (4.97) is simplified to

$$\frac{1}{2}\omega_{ij}^{\alpha\beta(\mathsf{k})}\|X-X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}+\big\langle\nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}),X-X^{(\mathsf{k})}\big\rangle+F_{ij}^{\alpha\beta}(X^{(\mathsf{k})})\geq F_{ij}^{\alpha\beta}(X).$$

The proof is completed.

4.12.2. Proof of Proposition 4.6.1

From Eqs. (4.26) and (4.27), we obtain

$$E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})})\geq\frac{1}{2}\omega_{ij}^{\alpha\beta(\mathsf{k})}\big\|X-X^{(\mathsf{k})}\big\|^2_{M_{ij}^{\alpha\beta}}+\big\langle\nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}),X-X^{(\mathsf{k})}\big\rangle+F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}),\tag{4.98}$$

where the equality "=" holds as long as $X=X^{(\mathsf{k})}$. From Proposition 4.5.1, we obtain

$$\frac{1}{2}\omega_{ij}^{\alpha\beta(\mathsf{k})}\|X-X^{(\mathsf{k})}\|^2_{M_{ij}^{\alpha\beta}}+\big\langle\nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}),X-X^{(\mathsf{k})}\big\rangle+F_{ij}^{\alpha\beta}(X^{(\mathsf{k})})\geq F_{ij}^{\alpha\beta}(X).\tag{4.99}$$

Then, as a result of Eqs. (4.98) and (4.99), it is straightforward to show

$$E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})})\geq F_{ij}^{\alpha\beta}(X)$$

for any $X \in \mathbb{R}^{d\times(d+1)n}$, where the equality "=" holds as long as $X = X^{(\mathsf{k})}$. The proof is completed.

4.12.3. Proof of Proposition 4.6.2

Proof of (a). From Eq. (4.25), it can be concluded that

$$\begin{aligned}(4.100)\quad \frac{1}{2}\|X - X^{(\mathsf{k})}\|^2_{\Omega_{ij}^{\alpha\beta}} = \kappa_{ij}^{\alpha\beta}\|R_i^\alpha - R_i^{\alpha(\mathsf{k})}\|^2 + \\ \tau_{ij}^{\alpha\beta}\|(R_i^\alpha - R_i^{\alpha(\mathsf{k})})\tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha - t_i^{\alpha(\mathsf{k})}\|^2 + \\ \kappa_{ij}^{\alpha\beta}\|R_j^\beta - R_j^{\beta(\mathsf{k})}\|^2 + \tau_{ij}^{\alpha\beta}\|t_j^\beta - t_j^{\beta(\mathsf{k})}\|^2.\end{aligned}$$

From Eq. (4.13), it is by definition that $F_{ij}^{\alpha\beta}(X)$ is a function related with $X^\alpha \in \mathcal{X}^\alpha$ and $X^\beta \in \mathcal{X}^\beta$ only, and thus, $\nabla F_{ij}^{\alpha\beta}(X)$ is sparse, which suggests

$$\begin{aligned}(4.101)\quad \left\langle \nabla F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), X - X^{(\mathsf{k})}\right\rangle = \left\langle \nabla_{X^\alpha} F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), X^\alpha - X^{\alpha(\mathsf{k})}\right\rangle + \\ \left\langle \nabla_{X^\beta} F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), X^\beta - X^{\beta(\mathsf{k})}\right\rangle.\end{aligned}$$

In Eq. (4.101), $\nabla_{X^\alpha} F_{ij}^{\alpha\beta}(X^{(\mathsf{k})})$ is the Euclidean gradient of $F_{ij}^{\alpha\beta}(X)$ with respect to $X^\alpha \in \mathcal{X}^\alpha$ at $X^{(\mathsf{k})} \in \mathcal{X}$. Substituting Eqs. (4.100) and (4.101) into Eq. (4.27), we obtain

$$\begin{aligned}(4.102)\quad & E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})}) \\ =&\omega_{ij}^{\alpha\beta(\mathsf{k})}\cdot\Big(\kappa_{ij}^{\alpha\beta}\|R_i^\alpha - R_i^{\alpha(\mathsf{k})}\|^2 + \tau_{ij}^{\alpha\beta}\|(R_i^\alpha - R_i^{\alpha(\mathsf{k})})\tilde{t}_{ij}^{\alpha\beta} + t_i^\alpha - t_i^{\alpha(\mathsf{k})}\|^2 + \\ & \kappa_{ij}^{\alpha\beta}\|R_j^\beta - R_j^{\beta(\mathsf{k})}\|^2 + \tau_{ij}^{\alpha\beta}\|t_j^\beta - t_j^{\beta(\mathsf{k})}\|^2\Big) + \left\langle \nabla_{X^\alpha} F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), X^\alpha - X^{\alpha(\mathsf{k})}\right\rangle + \\ & \left\langle \nabla_{X^\beta} F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}), X^\beta - X^{\beta(\mathsf{k})}\right\rangle + F_{ij}^{\alpha\beta}(X^{(\mathsf{k})}).\end{aligned}$$

In a similar way, $F_{ij}^{\alpha\alpha}(X)$ in Eq. (4.13) can be rewritten as

$$
\begin{aligned}
(4.103)\quad F_{ij}^{\alpha\alpha}(X) = \frac{1}{2}\kappa_{ij}^{\alpha\alpha}\|(R_i^\alpha - R_i^{\alpha(\mathsf{k})})\widetilde{R}_{ij}^{\alpha\alpha} - (R_j^\alpha - R_j^{\alpha(\mathsf{k})})\|^2 + \\
\frac{1}{2}\tau_{ij}^{\alpha\alpha}\|(R_i^\alpha - R_i^{\alpha(\mathsf{k})})\tilde{t}_{ij}^{\alpha\alpha} + t_i^\alpha - t_i^{\alpha(\mathsf{k})} - (t_j^\alpha - t_j^{\alpha(\mathsf{k})})\|^2 + \\
\left\langle \nabla_{X^\alpha} F_{ij}^{\alpha\alpha}(X^{(\mathsf{k})}), X^\alpha - X^{\alpha(\mathsf{k})} \right\rangle + F_{ij}^{\alpha\alpha}(X^{(\mathsf{k})}).
\end{aligned}
$$

Substituting Eqs. (4.102) and (4.103) into Eq. (4.29) and simplifying the resulting equation with Eq. (4.43), we obtain

$$
G(X|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} G^\alpha(X^\alpha|X^{(\mathsf{k})}) + F(X^{(\mathsf{k})}),
$$

where $G^\alpha(X^\alpha|X^{(\mathsf{k})})$ is a function that is related with $X^\alpha \in \mathcal{X}^\alpha$ only. Furthermore, a tedious but straightforward mathematical manipulation from Eqs. (4.101) to (4.103) indicates that there exists positive-semidefinite matrices $\Gamma^{\alpha(\mathsf{k})} \in \mathbb{R}^{(d+1)n_\alpha \times (d+1)n_\alpha}$ such that $G^\alpha(X^\alpha|X^{(\mathsf{k})})$ in the equation above can be written as

$$
G^\alpha(X^\alpha|X^{(\mathsf{k})}) = \frac{1}{2}\|X^\alpha - X^{\alpha(\mathsf{k})}\|^2_{\Gamma^{\alpha(\mathsf{k})}} + \left\langle \nabla_{X^\alpha} F(X^{(\mathsf{k})}),\, X^\alpha - X^{\alpha(\mathsf{k})} \right\rangle,
$$

where the formulation of $\Gamma^{\alpha(\mathsf{k})}$ is given in Section 4.6.3. The proof is completed.

Proof of (b). If we substitute Eq. (4.31) into Eq. (4.30), the result is

$$
(4.104)\quad G(X|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} \left[\frac{1}{2}\|X^\alpha - X^{\alpha(\mathsf{k})}\|^2_{\Gamma^{\alpha(\mathsf{k})}} + \left\langle \nabla_{X^\alpha} F(X^{(\mathsf{k})}),\, X^\alpha - X^{\alpha(\mathsf{k})} \right\rangle\right] + F(X^{(\mathsf{k})}).
$$

Furthermore, it can be shown that

$$\frac{1}{2}\|X - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}} = \sum_{\alpha\in\mathcal{A}} \frac{1}{2}\|X^\alpha - X^{\alpha(\mathsf{k})}\|^2_{\Gamma^{\alpha(\mathsf{k})}},$$

where $\Gamma^{(\mathsf{k})} \in \mathbb{R}^{(d+1)n\times(d+1)n}$ is defined as Eq. (4.33), and

$$\left\langle \nabla F(X^{(\mathsf{k})}), X - X^{(\mathsf{k})} \right\rangle = \sum_{\alpha\in\mathcal{A}} \left\langle \nabla_{X^\alpha} F(X^{(\mathsf{k})}), X^\alpha - X^{\alpha(\mathsf{k})} \right\rangle.$$

Thus, Eq. (4.104) is equivalent to Eq. (4.32), i.e.,

$$G(X|X^{(\mathsf{k})}) = \frac{1}{2}\|X - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}} + \left\langle \nabla F(X^{(\mathsf{k})}), X - X^{(\mathsf{k})} \right\rangle + F(X^{(\mathsf{k})}). \tag{4.105}$$

From Proposition 4.6.1, it is known that $E^{\alpha\beta}_{ij}(X|X^{(\mathsf{k})})$ majorizes $F^{\alpha\beta}_{ij}(X)$ and $E^{\alpha\beta}_{ij}(X|X^{(\mathsf{k})}) = F^{\alpha\beta}_{ij}(X^{(\mathsf{k})})$ if $X = X^{(\mathsf{k})}$. Then, as a result Eqs. (4.12) and (4.29), it can be concluded that $G(X|X^{(\mathsf{k})})$ majorizes $F(X)$ and $G(X|X^{(\mathsf{k})}) = F(X)$ if $X = X^{(\mathsf{k})}$. The proof is completed.

Proof of (c). From Eqs. (4.16), (4.27), (4.29) and (4.105), we rewrite $\Gamma^{(\mathsf{k})} \in \mathbb{R}^{(d+1)n\times(d+1)n}$ as

$$\Gamma^{(\mathsf{k})} = \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} M^{\alpha\alpha}_{ij} + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} \omega^{\alpha\beta(\mathsf{k})}_{ij} \cdot \Omega^{\alpha\beta}_{ij} + \xi\cdot\mathbf{I}, \tag{4.106}$$

where $\Omega^{\alpha\beta}_{ij} \succeq M^{\alpha\beta}_{ij}$ by Eq. (4.26) and $\xi \geq 0$. Then, as a result of Eqs. (4.21), (4.26) and (4.106), it is straightforward to conclude that

$$\Gamma^{(\mathsf{k})} \succeq M^{(\mathsf{k})} + \xi\cdot\mathbf{I} \succeq M^{(\mathsf{k})}. \tag{4.107}$$

The proof is completed.

Proof of (d). Let $\Gamma \in \mathbb{R}^{(d+1)n\times(d+1)n}$ be defined as

$$\Gamma \triangleq \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} M_{ij}^{\alpha\alpha} + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} \Omega_{ij}^{\alpha\beta} + \xi\cdot\mathbf{I}. \tag{4.108}$$

From Assumption 4.2(d) and Eq. (4.19), it can be concluded that

$$0 \leq \omega_{ij}^{\alpha\beta(\mathsf{k})} \leq 1 \tag{4.109}$$

for any $X^{(\mathsf{k})} \in \mathbb{R}^{d\times(d+1)n}$. Furthermore, it is known that $\Omega_{ij}^{\alpha\beta} \succeq 0$, then Eqs. (4.106), (4.108) and (4.109) result in $\Gamma \succeq \Gamma^{(\mathsf{k})}$ for any $X^{(\mathsf{k})} \in \mathbb{R}^{d\times(d+1)n}$. The proof is completed.

4.12.4. Proof of Proposition 4.7.1

Proof of (a). From Assumption 4.3 and line 8 of Algorithm 9, we obtain

$$G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) \leq G^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \tag{4.110}$$

and

$$H^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq H^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}). \tag{4.111}$$

From Eqs. (4.30), (4.36), (4.110) and (4.111), it can be concluded that

$$G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \leq G(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \tag{4.112}$$

and

$$H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq H(X^{(\mathsf{k})}|X^{(\mathsf{k})}). \tag{4.113}$$

Note that Eq. (4.42) suggests

$$F(X^{(\mathsf{k}+1)}) \leq G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \tag{4.114}$$

and

$$G(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}). \tag{4.115}$$

Then, Eqs. (4.112) to (4.115) result in

$$\begin{aligned} F(X^{(\mathsf{k}+1)}) \leq G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \leq G(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq \\ H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq H(X^{(\mathsf{k})}|X^{(\mathsf{k})}) = F(X^{(\mathsf{k})}), \end{aligned} \tag{4.116}$$

which indicates that $F(X^{(\mathsf{k})})$ is nonincreasing. The proof is completed.

Proof of (b). From Proposition 4.7.1(a), it has been proved that $F(X^{(\mathsf{k})})$ is nonincreasing. From Eq. (4.11) and Assumption 4.2, $F(X^{(\mathsf{k})}) \geq 0$, i.e., $F(X^{(\mathsf{k})})$ is bounded below. As a result, there exists $F^{\infty} \in \mathbb{R}$ such that $F(X^{(\mathsf{k})}) \to F^{\infty}$. The proof is completed.

Proof of (c). From Eq. (4.116), it is known that $F(X^{(\mathsf{k})}) \geq G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})})$, which suggests

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}). \tag{4.117}$$

From Eqs. (4.20) and (4.32), we obtain

$$F(X^{(\mathsf{k}+1)}) \leq \frac{1}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{M^{(\mathsf{k})}} + \left\langle \nabla F(X^{(\mathsf{k})}), X - X^{(\mathsf{k})} \right\rangle + F(X^{(\mathsf{k})}) \tag{4.118}$$

and

$$G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) = \frac{1}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}} + \left\langle \nabla F(X^{(\mathsf{k})}), X - X^{(\mathsf{k})} \right\rangle + F(X^{(\mathsf{k})}), \tag{4.119}$$

respectively. Substituting Eqs. (4.118) and (4.119) into the right-hand side of Eq. (4.117), we obtain

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq \frac{1}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}} - \frac{1}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{M^{(\mathsf{k})}}. \tag{4.120}$$

From Eqs. (4.107) and (4.120), there exists a constant scalar $\delta > 0$ such that

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq \frac{\delta}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2 \tag{4.121}$$

as long as $\xi > 0$. From Proposition 4.7.1(b), we obtain

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \to 0, \tag{4.122}$$

and thus, it can be concluded from Eqs. (4.121) and (4.122) that

$$\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0. \tag{4.123}$$

The proof is completed.

Proof of (d). From Eq. (4.116), it is known that $F(X^{(\mathsf{k})}) \geq H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})})$ and $G(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \geq G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \geq F(X^{(\mathsf{k}+1)})$, which suggests

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) - G(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}).$$

From Eqs. (4.32) and (4.40), the equation above is equivalent to

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq \frac{1}{2}\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2_{\Pi^{(\mathsf{k})}} - \frac{1}{2}\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}}. \tag{4.124}$$

A similar procedure to the derivation of Eq. (4.107) results in

$$\Pi^{(\mathsf{k})} \succeq \Gamma^{(\mathsf{k})} + (\zeta - \xi) \cdot \mathbf{I}, \tag{4.125}$$

which suggests there exists a constant scalar $\delta' > 0$ such that

$$\Pi^{(\mathsf{k})} \succeq \Gamma^{(\mathsf{k})} + \delta' \cdot \mathbf{I}$$

if $\zeta > \xi > 0$. Then, similar to the proof of Proposition 4.7.1(c), we obtain

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq \frac{\delta'}{2}\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2. \tag{4.126}$$

Thus, it can be concluded that

$$\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \to 0. \tag{4.127}$$

The proof is completed.

Proof of (e). The following proposition about $\nabla F(X)$ is needed in this proof.

Proposition 4.12.1. If Assumption 4.2(e) holds, then the Euclidean gradient $\nabla F(\cdot) : \mathbb{R}^{d\times(d+1)n} \to \mathbb{R}^{d\times(d+1)n}$ of $F(X)$ in Eq. (4.12) is Lipschitz continuous, i.e., there exists a constant $\mu > 0$ such that $\|\nabla F(X) - \nabla F(X')\| \leq \mu \cdot \|X - X'\|$.

Proof. From Assumption 4.2(e), it is known that $\rho(\|X\|^2)$ has Lipschitz continuous gradient, which suggests that $F_{ij}^{\alpha\beta}(X) = \dfrac{1}{2}\rho(\|X\|^2_{M_{ij}^{\alpha\beta}})$ in Eq. (4.17) has Lipschitz continuous gradient. Note that $F_{ij}^{\alpha\alpha}(X) = \dfrac{1}{2}\|X\|^2_{M_{ij}^{\alpha\alpha}}$ in Eq. (4.16) has Lipschitz continuous gradient as well. Then, from Eq. (4.12), it can be concluded that $F(X)$ has Lipschitz continuous gradient. The proof is completed. □

It is straightforward to show that the Riemannian gradient $\operatorname{grad} F(X)$ takes the form as

$$\operatorname{grad} F(X) = \begin{bmatrix} \operatorname{grad}_1 F(X) & \cdots & \operatorname{grad}_{|\mathcal{A}|} F(X) \end{bmatrix} \in T_X\mathcal{X}.$$

In the equation above, $\operatorname{grad}_\alpha F(X)$ is the Riemannian gradient of $F(X)$ with respect to $X^\alpha \in \mathcal{X}^\alpha$ for node $\alpha \in \mathcal{A}$, and can be written as

$$\operatorname{grad}_\alpha F(X) = \begin{bmatrix} \operatorname{grad}_{t^\alpha} F(X) & \operatorname{grad}_{R^\alpha} F(X) \end{bmatrix} \in T_{X^\alpha}\mathcal{X}^\alpha \tag{4.128}$$

where recall that

$$T_{X^\alpha}\mathcal{X}^\alpha \triangleq \mathbb{R}^{d\times n_\alpha} \times T_{R^\alpha} SO(d)^{n_\alpha}.$$

From [**7**, **90**], it can be shown that $\operatorname{grad}_{t^\alpha} F(X)$ and $\operatorname{grad}_{R^\alpha} F(X)$ in Eq. (4.128) are

$$\operatorname{grad}_{t^\alpha} F(X) = \nabla_{t^\alpha} F(X) \tag{4.129}$$

and

$$\text{grad}_{R^\alpha} F(X) = \nabla_{R^\alpha} F(X) - R^\alpha \,\text{SymBlockDiag}_d^\alpha(R^{\alpha\top} \nabla_{R^\alpha} F(X)). \tag{4.130}$$

In Eq. (4.130), $\text{SymBlockDiag}_d^\alpha : \mathbb{R}^{dn_\alpha \times dn_\alpha} \to \mathbb{R}^{dn_\alpha \times dn_\alpha}$ is a linear operator

$$\text{SymBlockDiag}_d^\alpha(Z) \triangleq \frac{1}{2}\text{BlockDiag}_d^\alpha(Z + Z^\top), \tag{4.131}$$

where $\text{BlockDiag}_d^\alpha : \mathbb{R}^{dn_\alpha \times dn_\alpha} \to \mathbb{R}^{dn_\alpha \times dn_\alpha}$ extracts the $d \times d$-block diagonals of a matrix, i.e.,

$$\text{BlockDiag}_d^\alpha(Z) \triangleq \begin{bmatrix} Z_{11} & & \\ & \ddots & \\ & & Z_{n_\alpha n_\alpha} \end{bmatrix} \in \mathbb{R}^{dn_\alpha \times dn_\alpha}.$$

As a result of Eqs. (4.128) to (4.131), there exists a linear operator

$$\mathcal{Q}_X : \mathbb{R}^{d \times d(n+1)} \to \mathbb{R}^{d \times d(n+1)} \tag{4.132}$$

that continuously depends on $X \in \mathcal{X}$ such that

$$\text{grad}\, F(X) = \mathcal{Q}_X(\nabla F(X)). \tag{4.133}$$

From Eq. (4.40), it is straightforward to show that

$$\begin{aligned} \nabla H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) &= \nabla F(X^{(\mathsf{k})}) + (X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})})\Pi^{(\mathsf{k})} = \\ &\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) + (X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})})\Pi^{(\mathsf{k})} + \big(\nabla F(X^{(\mathsf{k})}) - \nabla F(X^{(\mathsf{k}+\frac{1}{2})})\big). \end{aligned} \tag{4.134}$$

Note that Eq. (4.133) applies to any functions on $\mathcal{X}$. As a result of Eqs. (4.133) and (4.134), we obtain

$$\begin{aligned}(4.135)\quad \operatorname{grad} H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) = \operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})}) + \\ \mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big((X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})})\Pi^{(\mathsf{k})}\big) + \mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big(\nabla F(X^{(\mathsf{k})}) - \nabla F(X^{(\mathsf{k}+\frac{1}{2})})\big).\end{aligned}$$

From line 8 of Algorithm 9, we obtain.

$$\operatorname{grad} H^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) = \mathbf{0}.$$

In addition, it is by definition that

$$\operatorname{grad} H(X|X^{(\mathsf{k})}) = \Big[\operatorname{grad} H^{1}(X^{1}|X^{(\mathsf{k})}) \quad \cdots \quad \operatorname{grad} H^{|\mathcal{A}|}(X^{|\mathcal{A}|}|X^{(\mathsf{k})}),\Big]$$

which suggests

$$(4.136)\qquad \operatorname{grad} H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) = \mathbf{0}.$$

From Eqs. (4.135) and (4.136), we obtain

$$\operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})}) = \mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big((X^{(\mathsf{k})} - X^{(\mathsf{k}+\frac{1}{2})})\Pi^{(\mathsf{k})}\big) + \mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big(\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla F(X^{(\mathsf{k})})\big).$$

From the equation above, it can be shown that

$$
\begin{aligned}
&\|\operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})})\| \\
=&\big\|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big((X^{(\mathsf{k})} - X^{(\mathsf{k}+\frac{1}{2})})\Pi^{(\mathsf{k})}\big) + \mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big(\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla F(X^{(\mathsf{k})})\big)\big\| \\
\leq&\big\|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big((X^{(\mathsf{k})} - X^{(\mathsf{k}+\frac{1}{2})})\Pi^{(\mathsf{k})}\big)\big\| + \big\|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\big(\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla F(X^{(\mathsf{k})})\big)\big\| \\
\leq&\|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\Pi^{(\mathsf{k})}\|_2 \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \\
&\|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla F(X^{(\mathsf{k})})\|,
\end{aligned} \tag{4.137}
$$

where $\|\cdot\|_2$ denotes the induced 2-norm of linear operators. From Propositions 4.6.3(d) and 4.12.1, there exists a constant positive-semidefinite matrix $\Pi \in \mathbb{R}^{(d+1)n\times(d+1)n}$ and constant positive scalar $\mu > 0$ such that $\Pi \succeq \Pi^{(\mathsf{k})} \succeq 0$ and $\|\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla F(X^{(\mathsf{k})})\| \leq \mu \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|$ for any $\mathsf{k} \geq 0$, making it possible to upper-bound the right-hand side of Eq. (4.137):

(4.138)

$$
\|\operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq \|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\Pi\|_2 \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \mu \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|.
$$

Moreover, Eqs. (4.128) to (4.130) indicate that $\mathcal{Q}_X(\cdot)$ only depends on the rotation $R^\alpha \in SO(d)^{n_\alpha}$ for $\alpha \in \mathcal{A}$. Since $\mathcal{Q}_X(\cdot)$ is continuous and $SO(d)^{n_\alpha}$ is a compact manifold, $\|\mathcal{Q}_{X^{(\mathsf{k}+\frac{1}{2})}}\|_2$ is bounded for any $X^{(\mathsf{k}+\frac{1}{2})} \in \mathcal{X}$. Thus, there exists a constant scalar $\nu > 0$ such that the right-hand side of Eq. (4.138) can be upper-bounded as

$$
\|\operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq \nu\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|. \tag{4.139}
$$

As long as $\zeta > \xi > 0$, Eqs. (4.126) and (4.139) result in

$$\|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2 \leq \frac{2\nu^2}{\delta'}\big(F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)})\big).$$

Then, there exists a constant scalar $\epsilon = \frac{\delta'}{\nu^2} > 0$ with which the equation above can be rewritten as

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq \frac{\epsilon}{2}\|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2. \tag{4.140}$$

As a result of Eq. (4.140), we obtain

(4.141)

$$F(X^{(0)}) - F(X^{(\mathsf{K}+1)}) \geq \frac{\epsilon}{2}\sum_{\mathsf{k}=0}^{\mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2 \geq \frac{\epsilon(\mathsf{K}+1)}{2} \min_{0\leq \mathsf{k}\leq \mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2.$$

From Propositions 4.7.1(a) and 4.7.1(b), it can be concluded that $F(X^{(\mathsf{k}+1)}) \geq F^\infty$ for any $\mathsf{k} \geq 0$, which and Eq. (4.141) suggest

$$\min_{0\leq \mathsf{k}<\mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq \sqrt{\frac{2}{\epsilon}\cdot\frac{F(X^{(0)}) - F^\infty}{\mathsf{K}+1}}.$$

The proof is completed.

Proof of (f). As a result of Propositions 4.7.1(c) and 4.7.1(d), it is known that

$$\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0$$

and

$$\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \to 0$$

as long as $\zeta > \xi > 0$. Thus, it can be concluded from Eq. (4.139) that

$$\operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})}) \to \mathbf{0}$$

if $\zeta > \xi > 0$. In addition, Assumption 4.2(b) indicates that $\operatorname{grad} F(X^{(\mathsf{k})})$ is continuous, which suggests

$$\operatorname{grad} F(X^{(\mathsf{k})}) \to \operatorname{grad} F(X^{(\mathsf{k}+\frac{1}{2})}).$$

Then, we obtain

$$\operatorname{grad} F(X^{(\mathsf{k})}) \to \mathbf{0}.$$

The proof is completed.

4.12.5. Proof of Proposition 4.8.1

Proof of (a). In this proof, we will prove $F(X^{(\mathsf{k})}) \leq \overline{F}^{(\mathsf{k})}$ and $\overline{F}^{(\mathsf{k}+1)} \leq \overline{F}^{(\mathsf{k})}$ by induction.

1) From lines 4, 8, 14 of Algorithm 11, it can be shown that

$$F(X^{(-1)}) = F(X^{(0)}) = \overline{F}^{(-1)} = \overline{F}^{(0)}. \tag{4.142}$$

2) Suppose $\mathsf{k} \geq 0$ and $F(X^{(\mathsf{k})}) \leq \overline{F}^{(\mathsf{k})}$ holds at k-th iteration. In terms of $X^{(\mathsf{k}+\frac{1}{2})}$, if the adaptive restart scheme for $X^{(\mathsf{k}+\frac{1}{2})}$ is not triggered, it is immediate to show from line 10 of Algorithm 12 that

$$F(X^{(\mathsf{k}+\frac{1}{2})}) \leq \overline{F}^{(\mathsf{k})}. \tag{4.143}$$

On the other hand, if the adaptive restart scheme for $X^{(\mathsf{k}+\frac{1}{2})}$ is triggered, line 12 of Algorithm 12 results in

$$H^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq H^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}) = 0, \tag{4.144}$$

where $H^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}) = 0$ is from Eq. (4.38). Then, Eqs. (4.36), (4.42) and (4.144) indicate

$$F(X^{(\mathsf{k}+\frac{1}{2})}) \leq H(X^{(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq F(X^{(\mathsf{k})}) \leq \overline{F}^{(\mathsf{k})}. \tag{4.145}$$

Therefore, no matter whether the adaptive restart scheme is triggered or not, we conclude from Eqs. (4.143) and (4.145) that

$$F(X^{(\mathsf{k}+\frac{1}{2})}) \leq \overline{F}^{(\mathsf{k})} \tag{4.146}$$

always holds.

Furthermore, as a result of lines 25 to 27 of Algorithm 12, we obtain

$$F(X^{(\mathsf{k}+1)}) - \overline{F}^{(\mathsf{k})} \leq \phi \cdot \left(F(X^{(\mathsf{k}+\frac{1}{2})}) - \overline{F}^{(\mathsf{k})}\right) \leq 0. \tag{4.147}$$

From line 14 of Algorithm 11 and $F(X^{(\mathsf{k}+1)}) - \overline{F}^{(\mathsf{k})} \leq 0$ in Eq. (4.147), we obtain

$$F(X^{(\mathsf{k}+1)}) - \overline{F}^{(\mathsf{k}+1)} = (1-\eta) \cdot \left(F(X^{(\mathsf{k}+1)}) - \overline{F}^{(\mathsf{k})}\right) \leq 0$$

and

$$\overline{F}^{(\mathsf{k}+1)} - \overline{F}^{(\mathsf{k})} = \eta \cdot \left(F(X^{(\mathsf{k}+1)}) - \overline{F}^{(\mathsf{k})}\right) \leq 0,$$

which suggest $F(X^{(\mathsf{k}+1)}) \leq \overline{F}^{(\mathsf{k}+1)} \leq \overline{F}^{(\mathsf{k})}$.

3) Therefore, it can be concluded that $F(X^{(\mathsf{k})}) \leq \overline{F}^{(\mathsf{k})}$ and $\overline{F}^{(\mathsf{k}+1)} \leq \overline{F}^{(\mathsf{k})}$, which suggests that $\overline{F}^{(\mathsf{k})}$ is nonincreasing. The proof is completed.

Proof of (b). From line 14 of Algorithm 11, we obtain

$$\overline{F}^{(\mathsf{k}+1)} = (1-\eta)\cdot\overline{F}^{(\mathsf{k})} + \eta\cdot F(X^{(\mathsf{k}+1)}), \tag{4.148}$$

which suggests that $\overline{F}^{(\mathsf{k})}$ is a convex combination of $F(X^{(0)})$, $F(X^{(1)})$, $\cdots F(X^{(\mathsf{k})})$ as long as $\eta \in (0,\, 1]$. Since $F(X^{(\mathsf{k})}) \geq 0$ for any $\mathsf{k} \geq 0$, we obtain $\overline{F}^{(\mathsf{k})} \geq 0$ as well, i.e., $\overline{F}^{(\mathsf{k})}$ is bounded below. Proposition 4.8.1(a) indicates that $\overline{F}^{(\mathsf{k})}$ is nonincreasing, and thus, there exists F^{∞} such that $\overline{F}^{(\mathsf{k})} \to F^{\infty}$. Then, it can be still concluded from Eq. (4.148) that $F(X^{(\mathsf{k})}) \to F^{\infty}$.

Proof of (c). In terms of $X^{(\mathsf{k}+1)}$, there are three possible cases:

1) If $X^{(\mathsf{k}+1)}$ is from line 6 of Algorithm 12, then the adaptive restart scheme is not triggered and line 10 of Algorithm 12 results in

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \psi\cdot\left\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\right\|^2. \tag{4.149}$$

2) If $X^{(\mathsf{k}+1)}$ is from line 19 of Algorithm 12, then the adaptive restart scheme is triggered and Eq. (4.124) holds as well, from which and Eq. (4.107) we obtain

$$F(X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}) \geq \frac{\xi}{2}\left\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\right\|^2.$$

In the proof of Proposition 4.8.1(a), it is known that $\overline{F}^{(\mathsf{k})} \geq F(X^{(\mathsf{k})})$, then the equation above results in

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \frac{\xi}{2}\left\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\right\|^2. \tag{4.150}$$

3) If $X^{(\mathsf{k}+1)}$ is from line 26 of Algorithm 12, then we obtain $X^{(\mathsf{k}+1)} = X^{(\mathsf{k}+\frac{1}{2})}$ and $F(X^{(\mathsf{k}+1)}) = F(X^{(\mathsf{k}+\frac{1}{2})})$. Then, similar to the derivations of Eqs. (4.149) and (4.150), lines 5 and 12 of Algorithm 12 result in

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+\frac{1}{2})}) \geq \psi \cdot \left\| X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})} \right\|^2 \tag{4.151}$$

and

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+\frac{1}{2})}) \geq \frac{\zeta}{2} \left\| X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})} \right\|^2, \tag{4.152}$$

respectively, from which and $X^{(\mathsf{k}+1)} = X^{(\mathsf{k}+\frac{1}{2})}$ and $F(X^{(\mathsf{k}+1)}) = F(X^{(\mathsf{k}+\frac{1}{2})})$ we obtain either

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \psi \cdot \left\| X^{(\mathsf{k}+1)} - X^{(\mathsf{k})} \right\|^2$$

or

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \frac{\zeta}{2} \left\| X^{(\mathsf{k}+1)} - X^{(\mathsf{k})} \right\|^2.$$

Then, for any η, $\psi \in (0, 1]$, it can be shown from cases 1)) to 3)) above that there exists a constant scalar $\sigma > 0$ such that

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \frac{\sigma}{2} \| X^{(\mathsf{k}+1)} - X^{(\mathsf{k})} \|^2 \tag{4.153}$$

if $\xi > 0$ and $\zeta > 0$. In addition, note that Eq. (4.148) is equivalent to

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} = \eta \cdot \left(\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \right). \tag{4.154}$$

From Eqs. (4.153) and (4.154), we further conclude that there exists $\delta > 0$ such that

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} = \eta \cdot \left(\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)})\right) \geq \frac{\eta\sigma}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2 \geq \frac{\delta}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2. \tag{4.155}$$

Recall $\overline{F}^{(\mathsf{k})} \to F^{\infty}$ from Proposition 4.8.1(b), which suggests

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \to 0. \tag{4.156}$$

Therefore, Eqs. (4.155) and (4.156) indicate that

$$\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0.$$

The proof is completed.

Proof of (d). Note that Eqs. (4.151) and (4.152) suggest that there exists $\sigma' > 0$ such that

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+\frac{1}{2})}) \geq \frac{\sigma'}{2}\left\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\right\|^2 \tag{4.157}$$

always holds. From lines 25 to 27 of Algorithm 12, we obtain

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \phi \cdot \left(\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+\frac{1}{2})})\right) \geq \frac{\phi\sigma'}{2}\left\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\right\|^2, \tag{4.158}$$

where the last inequality is due to Eq. (4.157). From Eqs. (4.154) and (4.158), it can be shown that

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} = \eta \cdot \left(\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)})\right) \geq \frac{\eta\phi\sigma'}{2}\left\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\right\|^2. \tag{4.159}$$

The equation above suggests that there exists a constant scalar $\delta' > 0$ such that

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \geq \frac{\delta'}{2} \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2. \tag{4.160}$$

Note that Proposition 4.8.1(b) results in $\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \to 0$. Thus, similar to the proof of Proposition 4.8.1(c), it can be concluded from Eq. (4.160) that

$$\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \to 0$$

if $\zeta > \xi > 0$. The proof is completed.

Proof of (e). In terms of $X^{\alpha(\mathsf{k}+\frac{1}{2})} \in \mathcal{X}^\alpha$, there are two possible cases:

1) If $X^{\alpha(\mathsf{k}+\frac{1}{2})} \in \mathcal{X}^\alpha$ is from line 5 of Algorithm 12, we obtain

$$X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | Y^{(\mathsf{k})}). \tag{4.161}$$

From Eq. (4.38), we obtain

$$\begin{aligned}
&\nabla H^\alpha(X^{\alpha(\mathsf{k}+\frac{1}{2})}|Y^{(\mathsf{k})}) \\
=&\nabla_{X^\alpha} F(Y^{(\mathsf{k})}) + (X^{\alpha(\mathsf{k}+\frac{1}{2})} - Y^{\alpha(\mathsf{k})})\Pi^{\alpha(\mathsf{k})} \\
=&\nabla_{X^\alpha} F(X^{(\mathsf{k}+\frac{1}{2})}) + (X^{\alpha(\mathsf{k}+\frac{1}{2})} - Y^{\alpha(\mathsf{k})})\Pi^{\alpha(\mathsf{k})} + \big(\nabla_{X^\alpha} F(Y^{(\mathsf{k})}) - \nabla_{X^\alpha} F(X^{(\mathsf{k}+\frac{1}{2})})\big),
\end{aligned}$$

which suggests

$$\begin{aligned}
\operatorname{grad} H^\alpha(X^{\alpha(\mathsf{k}+\frac{1}{2})}|Y^{(\mathsf{k})}) = \operatorname{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})}) + \\
\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\big((X^{\alpha(\mathsf{k}+\frac{1}{2})} - Y^{\alpha(\mathsf{k})})\Pi^{\alpha(\mathsf{k})}\big) + \mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\big(\nabla_{X^\alpha} F(Y^{(\mathsf{k})}) - \nabla_{X^\alpha} F(X^{(\mathsf{k}+\frac{1}{2})})\big),
\end{aligned} \tag{4.162}$$

where $\mathrm{grad}_\alpha F(X)$ is the Riemannian gradient of $F(X)$ with respect to $X^\alpha \in \mathcal{X}^\alpha$, and $\mathcal{Q}^\alpha_{X^\alpha} : \mathbb{R}^{d\times dn_\alpha} \to \mathbb{R}^{d\times dn_\alpha}$ is a linear operator that extracts the α-th block of $\mathcal{Q}_X(\cdot)$ in Eq. (4.132). Since $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ is an optimal solution to Eq. (4.161), we obtain

$$\mathrm{grad}\, H^\alpha(X^{\alpha(\mathsf{k}+\frac{1}{2})}|Y^{(\mathsf{k})}) = \mathbf{0}. \tag{4.163}$$

From Eqs. (4.162) and (4.163), a straightforward mathematical manipulation indicates

$$\begin{aligned}\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})}) = \mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\big((Y^{\alpha(\mathsf{k})} - X^{\alpha(\mathsf{k}+\frac{1}{2})})\Pi^{\alpha(\mathsf{k})}\big)+\\ \mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\big(\nabla_{X^\alpha} F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla_{X^\alpha} F(Y^{(\mathsf{k})})\big).\end{aligned} \tag{4.164}$$

From Eq. (4.164), we further obtain

$$\begin{aligned}&\|\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})})\|\\ \le&\|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|(Y^{\alpha(\mathsf{k})} - X^{\alpha(\mathsf{k}+\frac{1}{2})})\Pi^{\alpha(\mathsf{k})}\| + \|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\nabla_{X^\alpha} F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla_{X^\alpha} F(Y^{(\mathsf{k})})\|\\ \le&\|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|(Y^{(\mathsf{k})} - X^{(\mathsf{k}+\frac{1}{2})})\Pi^{(\mathsf{k})}\| + \|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla F(Y^{(\mathsf{k})})\|\\ \le&\|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\Pi^{(\mathsf{k})}\|_2 \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - Y^{(\mathsf{k})}\| + \|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\nabla F(X^{(\mathsf{k}+\frac{1}{2})}) - \nabla F(Y^{(\mathsf{k})})\|,\end{aligned}$$

where $\|\cdot\|_2$ denotes the induced 2-norm of linear operators. From Propositions 4.6.3(d) and 4.12.1, the equation above results in

$$\begin{aligned}\|\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})})\| \le \|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \|\Pi\|_2 \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - Y^{(\mathsf{k})}\|+\\ \|\mathcal{Q}^\alpha_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2 \cdot \mu \cdot \|X^{(\mathsf{k}+\frac{1}{2})} - Y^{(\mathsf{k})}\|.\end{aligned} \tag{4.165}$$

Similar to Eqs. (4.138) and (4.139), $\|\mathcal{Q}^{\alpha}_{X^{\alpha(\mathsf{k}+\frac{1}{2})}}\|_2$ in Eq. (4.165) is bounded as well. Therefore, there exists a constant scalar $\nu^{\alpha} > 0$ such that the right-hand side of Eq. (4.165) can be upper-bounded:

$$\|\mathrm{grad}_{\alpha} F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq \nu^{\alpha} \|X^{(\mathsf{k}+\frac{1}{2})} - Y^{(\mathsf{k})}\|. \tag{4.166}$$

Recall that $Y^{(\mathsf{k})} \in \mathbb{R}^{d\times(d+1)n}$ results from line 12 of Algorithm 11:

$$Y^{(\mathsf{k})} = X^{(\mathsf{k})} + \big(X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\big)\lambda^{(\mathsf{k})}. \tag{4.167}$$

In Eq. (4.167), $\lambda^{(\mathsf{k})} \in \mathbb{R}^{(d+1)n\times(d+1)n}$ is a diagonal matrix

$$\lambda^{(\mathsf{k})} \triangleq \mathrm{diag}\{\lambda^{1(\mathsf{k})} \cdot \mathbf{I}^1, \cdots, \lambda^{|\mathcal{A}|(\mathsf{k})} \cdot \mathbf{I}^{|\mathcal{A}|}\} \in \mathbb{R}^{(d+1)n\times(d+1)n},$$

where $\lambda^{\alpha(\mathsf{k})} \in \mathbb{R}$ is given by line 11 of Algorithm 11 and $\mathbf{I}^{\alpha} \in \mathbb{R}^{(d+1)n_{\alpha}\times(d+1)n_{\alpha}}$ is the identity matrix. From Eqs. (4.166) and (4.167), it can be shown that

$$\begin{aligned}
&\|\mathrm{grad}_{\alpha} F(X^{(\mathsf{k}+\frac{1}{2})})\| \\
\leq&\nu^{\alpha}\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})} - \big(X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\big)\lambda^{(\mathsf{k})}\| \\
\leq&\nu^{\alpha}\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \nu^{\alpha}\|\big(X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\big)\lambda^{(\mathsf{k})}\| \\
\leq&\nu^{\alpha}\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \nu^{\alpha}\|\lambda^{(\mathsf{k})}\|_2 \cdot \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\|.
\end{aligned} \tag{4.168}$$

From line 11 of Algorithm 11, we obtain $s^{\alpha(\mathsf{k})} \geq 1$, and thus,

$$\lambda^{\alpha(\mathsf{k})} = \frac{\sqrt{4{s^{\alpha(\mathsf{k})}}^2 + 1} - 1}{2s^{\alpha(\mathsf{k})}} = \frac{2s^{\alpha(\mathsf{k})}}{\sqrt{4{s^{\alpha(\mathsf{k})}}^2 + 1} + 1} \in (0,\, 1),$$

which suggests $\|\lambda^{(\mathsf{k})}\|_2 \in (0,\, 1)$. Then, we upper-bound the right-hand side of Eq. (4.168) using $\|\lambda^{(\mathsf{k})}\|_2 \in (0,\, 1)$:

$$\|\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq \nu^\alpha \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \nu^\alpha \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\|. \tag{4.169}$$

2) If $X^{\alpha(\mathsf{k}+\frac{1}{2})} \in \mathcal{X}^\alpha$ is from line 12 of Algorithm 12, we obtain

$$X^{\alpha(\mathsf{k}+\frac{1}{2})} \leftarrow \arg\min_{X^\alpha \in \mathcal{X}^\alpha} H^\alpha(X^\alpha | X^{(\mathsf{k})}). \tag{4.170}$$

A similar procedure to the derivation of Eq. (4.166) indicates

$$\|\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq \nu^\alpha \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|,$$

where $\nu^\alpha > 0$ is the same as that in Eq. (4.166). Thus, we obtain

$$\|\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})}) \leq \nu^\alpha \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \leq \nu^\alpha \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \nu^\alpha \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\|.$$

Therefore, no matter if $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ is from line 5 or 12 of Algorithm 12, it can be shown that

$$\|\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq \nu^\alpha \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \nu^\alpha \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\| \tag{4.171}$$

holds for any node $\alpha \in \mathcal{A}$. From Eq. (4.171), there exists a constant scalar $\nu \triangleq \sum_{\alpha\in\mathcal{A}} \nu^\alpha > 0$ such that

$$\begin{aligned}
&\|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\| \\
\leq &\sum_{\alpha\in\mathcal{A}} \|\mathrm{grad}_\alpha F(X^{(\mathsf{k}+\frac{1}{2})})\| \\
\leq &\sum_{\alpha\in\mathcal{A}} \nu^\alpha \cdot \left(\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\|\right) \\
= &\nu \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| + \nu \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\| \\
\leq &\sqrt{2}\nu \sqrt{\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2 + \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\|^2}.
\end{aligned}$$

Furthermore, if $\zeta > \xi > 0$, the equation above indicates

$$\|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2 \leq 2\nu^2 \cdot \big(\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2 + \|X^{(\mathsf{k})} - X^{(\mathsf{k}-1)}\|^2\big) \leq \frac{4\nu^2}{\delta'}\big(\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)}\big) + \frac{4\nu^2}{\delta}\big(\overline{F}^{(\mathsf{k}-1)} - \overline{F}^{(\mathsf{k})}\big), \tag{4.172}$$

where the last inequality is due to Eqs. (4.155) and (4.160). Letting $\epsilon \triangleq \min\{\frac{\delta}{2\nu^2}, \frac{\delta'}{2\nu^2}\} > 0$, then Eq. (4.172) leads to

$$\overline{F}^{(\mathsf{k}-1)} - \overline{F}^{(\mathsf{k}+1)} \geq \frac{\epsilon}{2}\|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2. \tag{4.173}$$

A telescoping sum of Eq. (4.173) over k from 0 to K yields

$$\overline{F}^{(-1)} + \overline{F}^{(0)} - \overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \geq \frac{\epsilon}{2}\sum_{\mathsf{k}=0}^{\mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2,$$

and thus,

$$\overline{F}^{(-1)} + \overline{F}^{(0)} - \overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \geq \frac{\epsilon(\mathsf{K}+1)}{2} \min_{0\leq \mathsf{k} \leq \mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2. \tag{4.174}$$

From lines 4 and 8 of Algorithm 11, we obtain

$$\overline{F}^{(-1)} = \overline{F}^{(0)} = F(X^{(0)}), \tag{4.175}$$

and Propositions 4.8.1(a) and 4.8.1(b) indicate

$$\overline{F}^{(\mathsf{k})} \geq \overline{F}^{(\mathsf{k}+1)} \geq F^{\infty}. \tag{4.176}$$

As a result of Eqs. (4.174) to (4.176), it can be concluded that

$$F(X^{(0)}) - F^{\infty} \geq \frac{\epsilon(\mathsf{K}+1)}{4} \min_{0\leq \mathsf{k} \leq \mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\|^2,$$

which is equivalent to

$$\min_{0\leq \mathsf{k}<\mathsf{K}} \|\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})})\| \leq 2\sqrt{\frac{1}{\epsilon}\cdot\frac{F(X^{(0)})-F^{\infty}}{\mathsf{K}+1}}. \tag{4.177}$$

The proof is completed.

Proof of (f). From Propositions 4.8.1(c) and 4.8.1(d), we obtain

$$\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0$$

and

$$\|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\| \to 0$$

as long as $\zeta > \xi > 0$, from which and Eq. (4.172), it is trivial to show that

$$\mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})}) \to \mathbf{0}. \tag{4.178}$$

In addition, note that $\mathrm{grad}\, F(X)$ is continuous by Assumption 4.2(b), which suggests

$$\mathrm{grad}\, F(X^{(\mathsf{k})}) \to \mathrm{grad}\, F(X^{(\mathsf{k}+\frac{1}{2})}). \tag{4.179}$$

From Eqs. (4.178) and (4.179), it can be concluded that

$$\mathrm{grad}\, F(X^{(\mathsf{k})}) \to \mathbf{0}.$$

The proof is completed.

4.12.6. Proof of Proposition 4.9.1

Proof of (a). We will prove $F(X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k})}$ by induction.

1) From Eq. (4.12), it can be shown that

$$(4.180)\quad F(X^{(0)}) = \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} F_{ij}^{\alpha\alpha}(X^{(0)}) + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} F_{ij}^{\alpha\beta}(X^{(0)}) =$$
$$\sum_{\alpha\in\mathcal{A}} \Bigg(\sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} F_{ij}^{\alpha\alpha}(X^{(0)}) + \frac{1}{2} \sum_{\beta\in\mathcal{N}_-^{\alpha}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} F_{ij}^{\alpha\beta}(X^{(0)}) +$$
$$\frac{1}{2} \sum_{\beta\in\mathcal{N}_+^{\alpha}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\beta\alpha}} F_{ij}^{\beta\alpha}(X^{(0)}) \Bigg) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(0)},$$

where the last equality results from Eq. (4.81).

2) Suppose $F(X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k})}$ holds at k-th iteration. As a result of Eq. (4.83), we obtain

$$(4.181)\quad \sum_{\alpha\in\mathcal{A}} G^{\alpha(\mathsf{k}+1)} = \sum_{\alpha\in\mathcal{A}} G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) + \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k})}$$
$$= \sum_{\alpha\in\mathcal{A}} G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) + F(X^{(\mathsf{k})}) = G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}),$$

where the second and third equality are due to $F(X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k})}$ and Eq. (4.30), respectively. From Eq. (4.80) and

$$\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2 = \sum_{\alpha\in\mathcal{A}} \|X^{\alpha(\mathsf{k}+1)} - X^{\alpha(\mathsf{k})}\|^2,$$

it is straightforward to show

$$(4.182)\quad \sum_{\alpha\in\mathcal{A}} \Delta G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) =$$
$$\sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} \Big(F_{ij}^{\alpha\beta}(X) - E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k})}) \Big) - \frac{\xi}{2} \big\| X^{(\mathsf{k}+1)} - X^{(\mathsf{k})} \big\|^2.$$

In addition, Eqs. (4.84) and (4.181) suggest

$$\sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k}+1)} = \sum_{\alpha\in\mathcal{A}} G^{\alpha(\mathsf{k}+1)} + \sum_{\alpha\in\mathcal{A}} \Delta G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}) \\ = G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) + \sum_{\alpha\in\mathcal{A}} \Delta G^{\alpha}(X^{\alpha(\mathsf{k}+1)}|X^{(\mathsf{k})}). \tag{4.183}$$

Substituting Eqs. (4.29) and (4.182) into Eq. (4.183) yields.

$$\sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k}+1)} = \sum_{\alpha\in\mathcal{A}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\alpha}} F_{ij}^{\alpha\alpha}(X^{(\mathsf{k}+1)}) + \sum_{\substack{\alpha,\beta\in\mathcal{A},\\ \alpha\neq\beta}} \sum_{(i,j)\in\overrightarrow{\mathcal{E}}^{\alpha\beta}} F_{ij}^{\alpha\beta}(X^{(\mathsf{k}+1)}).$$

We simplify the equation above with Eq. (4.12) and obtain

$$F(X^{(\mathsf{k}+1)}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k}+1)}. \tag{4.184}$$

3) Therefore, it can be concluded that $F(X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k})}$ holds for any $\mathsf{k} \geq 0$.

Proof of (b). We will prove $\overline{F}^{(\mathsf{k})} = \sum_{\alpha\in\mathcal{A}} \overline{F}^{\alpha(\mathsf{k})}$ by induction.

1) Recall from Eqs. (4.79), (4.82) and (4.180) that $\overline{F}^{(0)} = F(X^{(0)})$, $\overline{F}^{\alpha(0)} = F^{\alpha(0)}$ and $F(X^{(0)}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(0)}$, which immediately yields

$$\overline{F}^{(0)} = F(X^{(0)}). \tag{4.185}$$

2) Suppose $\overline{F}^{(\mathsf{k})} = \sum_{\alpha\in\mathcal{A}} \overline{F}^{\alpha(\mathsf{k})}$ holds at k-th iteration. As a result of Eq. (4.79), we obtain

$$\overline{F}^{(\mathsf{k}+1)} = (1-\eta)\cdot\overline{F}^{(\mathsf{k})} + \eta\cdot F(X^{(\mathsf{k}+1)}).$$

Note that Proposition 4.9.1(a) suggests $F(X^{(\mathsf{k}+1)}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k}+1)}$. Apply $\overline{F}^{(\mathsf{k})} = \sum_{\alpha\in\mathcal{A}} \overline{F}^{\alpha(\mathsf{k})}$ and $F(X^{(\mathsf{k}+1)}) = \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k}+1)}$ on the right-hand side of the equation above results in

$$\overline{F}^{(\mathsf{k}+1)} = (1-\eta)\cdot\sum_{\alpha\in\mathcal{A}}\overline{F}^{\alpha(\mathsf{k})} + \eta\cdot\sum_{\alpha\in\mathcal{A}}F^{\alpha(\mathsf{k}+1)} = \sum_{\alpha\in\mathcal{A}}\overline{F}^{\alpha(\mathsf{k}+1)}, \tag{4.186}$$

where the last equality is due to Eq. (4.85).

3) Therefore, it can be concluded that $\overline{F}^{(\mathsf{k})} = \sum_{\alpha\in\mathcal{A}}\overline{F}^{\alpha(\mathsf{k})}$ holds for any $\mathsf{k}\geq 0$. The proof is completed.

Proof of (c). Proposition 4.6.1 indicates $E_{ij}^{\alpha\beta}(X|X^{(\mathsf{k}-1)}) \geq F_{ij}^{\alpha\beta}(X)$ and $E_{ij}^{\beta\alpha}(X|X^{(\mathsf{k}-1)}) \geq F_{ij}^{\beta\alpha}(X)$, from which and Eq. (4.80) we obtain

$$\Delta G^{\alpha}(X^{\alpha}|X^{(\mathsf{k}-1)}) \leq 0 \tag{4.187}$$

as long as $\xi \geq 0$. From Eqs. (4.84) and (4.187), it is immediate to conclude

$$F^{\alpha(\mathsf{k}+1)} \leq G^{\alpha(\mathsf{k}+1)} \tag{4.188}$$

for any $\mathsf{k}\geq 0$. If $G^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$, the equation above further suggests

$$F^{\alpha(\mathsf{k}+1)} \leq G^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}. \tag{4.189}$$

From Eq. (4.85), we obtain

$$\overline{F}^{\alpha(\mathsf{k}+1)} = (1-\eta)\cdot\overline{F}^{\alpha(\mathsf{k})} + \eta\cdot F^{\alpha(\mathsf{k}+1)}, \tag{4.190}$$

where note that $\eta\in(0,1]$. Thus, we conclude that $\overline{F}^{\alpha(\mathsf{k}+1)}$ is a convex combination of $\overline{F}^{\alpha(\mathsf{k})}$ and $F^{\alpha(\mathsf{k}+1)}$, which and Eq. (4.189) lead to

$$F^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}. \tag{4.191}$$

The proof is completed.

4.12.7. Proof of Proposition 4.9.2

Proof of (a). We will prove $F^{\alpha(\mathsf{k})} \leq \overline{F}^{\alpha(\mathsf{k})}$ and $\overline{F}^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$ by induction, from which it can be shown that $\overline{F}^{(\mathsf{k}+1)} \leq \overline{F}^{(\mathsf{k})}$.

1) From line 7 of Algorithm 13, it can be concluded that

$$F^{\alpha(0)} = \overline{F}^{\alpha(0)}.$$

2) Suppose $F^{\alpha(\mathsf{k})} \leq \overline{F}^{\alpha(\mathsf{k})}$ holds at k-th iteration. If the adaptive restart scheme for $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ is not triggered, it is immediate to show from line 7 of Algorithm 14 that

$$G^{\alpha(\mathsf{k}+\frac{1}{2})} \leq \overline{F}^{\alpha(\mathsf{k})}.$$

On the other hand, if the adaptive restart scheme for $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ is triggered, line 8 of Algorithm 14 results in

$$G^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq H^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) \leq H^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}) = 0, \tag{4.192}$$

where the first inequality and the last equality are due to Proposition 4.6.3(e) and Eq. (4.38), respectively. From Eqs. (4.83) and (4.192), we obtain

$$G^{\alpha(\mathsf{k}+\frac{1}{2})} = G^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) + F^{\alpha(\mathsf{k})} \leq H^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) + F^{\alpha(\mathsf{k})} \leq F^{\alpha(\mathsf{k})} \leq \overline{F}^{\alpha(\mathsf{k})}. \tag{4.193}$$

Therefore, no matter whether the adaptive restart scheme is triggered or not, it can be concluded that

$$G^{\alpha(\mathsf{k}+\frac{1}{2})} \leq \overline{F}^{\alpha(\mathsf{k})}. \tag{4.194}$$

In addition, line 16 of Algorithm 14 and Eq. (4.194) result in

$$G^{\alpha(\mathsf{k}+1)} - \overline{F}^{\alpha(\mathsf{k})} \leq \phi \cdot \left(G^{\alpha(\mathsf{k}+\frac{1}{2})} - \overline{F}^{\alpha(\mathsf{k})}\right) \leq 0, \tag{4.195}$$

which suggests

$$G^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}. \tag{4.196}$$

As a result of Proposition 4.9.1(c), the equation above indicates

$$F^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}. \tag{4.197}$$

3) Therefore, it can be concluded that $F^{\alpha(\mathsf{k})} \leq \overline{F}^{\alpha(\mathsf{k})}$ and $\overline{F}^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$ hold for any $\mathsf{k} \geq 0$.

4) Summing both sides of $\overline{F}^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$ over all the nodes α and implementing Propositions 4.9.1(a) and 4.9.1(b) yields

$$\overline{F}^{(\mathsf{k}+1)} = \sum_{\alpha \in \mathcal{A}} \overline{F}^{\alpha(\mathsf{k}+1)} \leq \sum_{\alpha \in \mathcal{A}} \overline{F}^{\alpha(\mathsf{k})} = \overline{F}^{(\mathsf{k})},$$

which suggests that $\overline{F}^{(\mathsf{k})}$ is nonincreasing. The proof is completed.

Proof of (b). Recalling that $F(X^{(\mathsf{k})}) \geq 0$ holds by definition for any $\mathsf{k} \geq 0$ and $\overline{F}^{(\mathsf{k})}$ is the exponential moving average of $F(X^{(0)})$, $F(X^{(1)})$, $\cdots$, $F(X^{(\mathsf{k})})$, we obtain $\overline{F}^{(\mathsf{k})} \geq 0$, i.e., $\overline{F}^{(\mathsf{k})}$ is bounded below. In addition, Proposition 4.9.2(a) indicates that $\overline{F}^{(\mathsf{k})}$ is nonincreasing, and thus, there exists F^{∞} such that $\overline{F}^{(\mathsf{k})} \to F^{\infty}$, from which and Eq. (4.79) it can be concluded that $F(X^{(\mathsf{k})}) \to F^{\infty}$ as well. The proof is completed.

Proof of (c). From Eq. (4.30), it can be shown that $G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})})$ takes the form as

(4.198)
$$G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} G^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}) + F(X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} G^{\alpha}(X^{\alpha(\mathsf{k})}|X^{(\mathsf{k})}) + \sum_{\alpha\in\mathcal{A}} F^{\alpha(\mathsf{k})},$$

where the last equality is due to Proposition 4.9.1(a). Applying Eq. (4.83) on the equation above results in

(4.199)
$$G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} G^{\alpha(\mathsf{k}+1)}.$$

Recalling $G^{\alpha(\mathsf{k}+1)} \leq \overline{F}^{\alpha(\mathsf{k})}$ from Eq. (4.196) and $\sum_{\alpha\in\mathcal{A}} \overline{F}^{\alpha(\mathsf{k})} = \overline{F}^{(\mathsf{k})}$ from Proposition 4.9.1(b), we obtain

(4.200)
$$G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) = \sum_{\alpha\in\mathcal{A}} G^{\alpha(\mathsf{k}+1)} \leq \sum_{\alpha\in\mathcal{A}} \overline{F}^{\alpha(\mathsf{k})} = \overline{F}^{(\mathsf{k})}.$$

From Eq. (4.200), it can be shown that

(4.201)
$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) - F(X^{(\mathsf{k}+1)}).$$

Substitute Eqs. (4.118) and (4.119) into the right-hand side of Eq. (4.201) and simplify the resulting equation:

(4.202)
$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \frac{1}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}} - \frac{1}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{M^{(\mathsf{k})}}.$$

From Eqs. (4.154) and (4.202), we obtain

(4.203)
$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} = \eta\cdot\left(\overline{F}^{(\mathsf{k})} - F^{(\mathsf{k}+1)}\right) \geq \frac{\eta}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{\Gamma^{(\mathsf{k})}} - \frac{\eta}{2}\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2_{M^{(\mathsf{k})}}.$$

From Eq. (4.107), note that $\Gamma^{(\mathsf{k})} - M^{(\mathsf{k})} \geq \xi \cdot \mathbf{I}$, and thus, there exists $\delta > 0$ such that the equation above is reduced to

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \geq \frac{\delta}{2} \|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\|^2 \tag{4.204}$$

as long as $\xi > 0$. Furthermore, Proposition 4.9.2(b) yields

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \to 0, \tag{4.205}$$

from which and Eq. (4.204), we obtain

$$\|X^{(\mathsf{k}+1)} - X^{(\mathsf{k})}\| \to 0. \tag{4.206}$$

The proof is completed.

Proof of (d). In terms of $X^{\alpha(\mathsf{k}+\frac{1}{2})}$, there are two possible cases:

1) If $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ is from line 3 of Algorithm 14, then line 7 of Algorithm 14 indicates

$$\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+\frac{1}{2})} \geq \psi \cdot \|X^{\alpha(\mathsf{k}+\frac{1}{2})} - X^{\alpha(\mathsf{k})}\|^2. \tag{4.207}$$

2) If $X^{\alpha(\mathsf{k}+\frac{1}{2})}$ is from line 8 of Algorithm 14, then note that Eq. (4.193) holds for any $\mathsf{k} \geq 0$, which suggests

$$\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+\frac{1}{2})} \geq H^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}) - G^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})}).$$

Recalling the definitions of $G^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})})$ and $H^{\alpha}(X^{\alpha(\mathsf{k}+\frac{1}{2})}|X^{(\mathsf{k})})$ in Eqs. (4.31) and (4.38), we rewrite the equation above as

$$\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+\frac{1}{2})} \geq \frac{1}{2}\|X^{\alpha(\mathsf{k}+\frac{1}{2})} - X^{\alpha(\mathsf{k})}\|^2_{\Pi^{\alpha(\mathsf{k})}} - \frac{1}{2}\|X^{\alpha(\mathsf{k}+\frac{1}{2})} - X^{\alpha(\mathsf{k})}\|^2_{\Gamma^{\alpha(\mathsf{k})}}. \tag{4.208}$$

Similar to $\Gamma^{(\mathsf{k})}$ and $\Pi^{(\mathsf{k})}$ in Eq. (4.125), we obtain

$$\Pi^{\alpha(\mathsf{k})} \succeq \Gamma^{\alpha(\mathsf{k})} + (\zeta - \xi) \cdot \mathbf{I}. \tag{4.209}$$

Applying Eq. (4.209) on the right-hand side of Eq. (4.208) indicates

$$\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+\frac{1}{2})} \geq \frac{\zeta - \xi}{2} \|X^{\alpha(\mathsf{k}+\frac{1}{2})} - X^{\alpha(\mathsf{k})}\|^2. \tag{4.210}$$

Then, as a result of Eqs. (4.207) and (4.210), there exists a constant scalar $\sigma' > 0$ such that

$$\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+\frac{1}{2})} \geq \frac{\sigma'}{2} \|X^{\alpha(\mathsf{k}+\frac{1}{2})} - X^{\alpha(\mathsf{k})}\|^2 \tag{4.211}$$

if $\zeta > \xi > 0$. In addition, lines 16 to 18 of Algorithm 14 results in

$$\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+1)} \geq \phi \cdot \left(\overline{F}^{\alpha(\mathsf{k})} - G^{\alpha(\mathsf{k}+\frac{1}{2})}\right) \geq \frac{\phi\sigma'}{2} \|X^{\alpha(\mathsf{k}+\frac{1}{2})} - X^{\alpha(\mathsf{k})}\|^2, \tag{4.212}$$

where the last inequality is from Eq. (4.211). Summing both sides of Eq. (4.212) over all the nodes $\alpha \in \mathcal{A}$ and simplifying the resulting equation with Proposition 4.9.1(b) and Eq. (4.199), we obtain

$$\overline{F}^{(\mathsf{k})} - G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \geq \frac{\phi\sigma'}{2} \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2.$$

Recalling $G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \geq F(X^{(\mathsf{k}+1)})$ from Proposition 4.6.2(b), the equation above indicates

$$\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \geq \overline{F}^{(\mathsf{k})} - G(X^{(\mathsf{k}+1)}|X^{(\mathsf{k})}) \geq \frac{\phi\sigma'}{2} \|X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})}\|^2. \tag{4.213}$$

From Eqs. (4.154) and (4.213), it is immediate to show

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \geq \eta \cdot \left(\overline{F}^{(\mathsf{k})} - F(X^{(\mathsf{k}+1)}) \right) \geq \frac{\eta\phi\sigma'}{2} \| X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})} \|^2.$$

Therefore, there exists a constant scalar $\delta' > 0$ such that

$$\overline{F}^{(\mathsf{k})} - \overline{F}^{(\mathsf{k}+1)} \geq \frac{\delta'}{2} \| X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})} \|^2. \tag{4.214}$$

Since $\overline{F}^{(\mathsf{k})} \to F^{\infty}$, it can be concluded that

$$\| X^{(\mathsf{k}+\frac{1}{2})} - X^{(\mathsf{k})} \| \to 0 \tag{4.215}$$

if $\zeta > \xi > 0$. The proof is completed.

Proof of (e) and (f). The proofs of Propositions 4.9.2(e) and 4.9.2(f) are almost the same as these of Propositions 4.8.1(e) and 4.8.1(f), which are thus omitted due to space limitation.

CHAPTER 5

Sparse Constrained Optimization of 3D Human Pose and Shape Estimation

We propose a novel sparse constrained formulation and from it derive a real-time optimization method for 3D human pose and shape estimation. Our optimization method, SCOPE (Sparse Constrained Optimization for 3D human Pose and shapE estimation), is orders of magnitude faster (avg. 4ms convergence) than existing optimization methods, while being mathematically equivalent to their dense unconstrained formulation under mild assumptions. We achieve this by exploiting the underlying sparsity and constraints of our formulation to efficiently compute the Gauss-Newton direction. We show that this computation scales linearly with the number of joints and measurements of a complex 3D human model, in contrast to prior work where it scales cubically due to their dense unconstrained formulation. Based on our optimization method, we present a real-time motion capture framework that estimates 3D human poses and shapes from a single image at over 30 FPS. In benchmarks against state-of-the-art methods on multiple public datasets, our framework outperforms other optimization methods and achieves competitive accuracy against regression methods.

5.1. Introduction

Estimating 3D human poses and shapes from an image has a broad range of applications in embodied AI, robotics, AR/VR, and has seen remarkable progress in recent years.

Among leading techniques, optimization methods [**11**,**12**,**137**–**140**] have been moderately successful. However, they can still take up to tens of seconds to fit 3D human poses and shapes given an image, which is not practical for real-time applications. Deep learning based regression methods [**2**,**141**] have significantly reduced the computation times down to just tens of milliseconds, but often rely on optimization during training or for refining the network outputs. With a novel formulation, we propose fast algorithms to solve the optimization problem of 3D human pose and shape estimation in real time and achieve a comparable performance to the regression methods [**2**, **141**].

Most optimization methods [**11**, **12**, **137**–**140**] formulate 3D human pose and shape estimation as dense unconstrained optimization problems, differing only in terms of the objective functions. These formulations are dense as they result in dense Hessian matrices and unconstrained as the optimization variables are unconstrained. To optimize the objective they use iterative techniques like Gauss-Newton [**142**] to find a local minimum given an initial guess. These formulations, however, suffer from high computation times due to the dense Hessian matrices that lead to $O(K^3) + O(K^2N)$ time to compute the Gauss-Newton direction for a 3D human model with K joints and N measurements. In particular, computing this descent direction involves the steps of linearization to find the Jacobian, building and then solving the linear system, where a dense formulation renders all these steps expensive. Therefore, it is critical to improve the efficiency of the Gauss-Newton direction computation to develop real-time optimization methods for 3D human pose and shape estimation.

The preliminary results of this work are presented in [**143**]. Instead of using the dense unconstrained formulation from existing optimization methods, we present a sparse

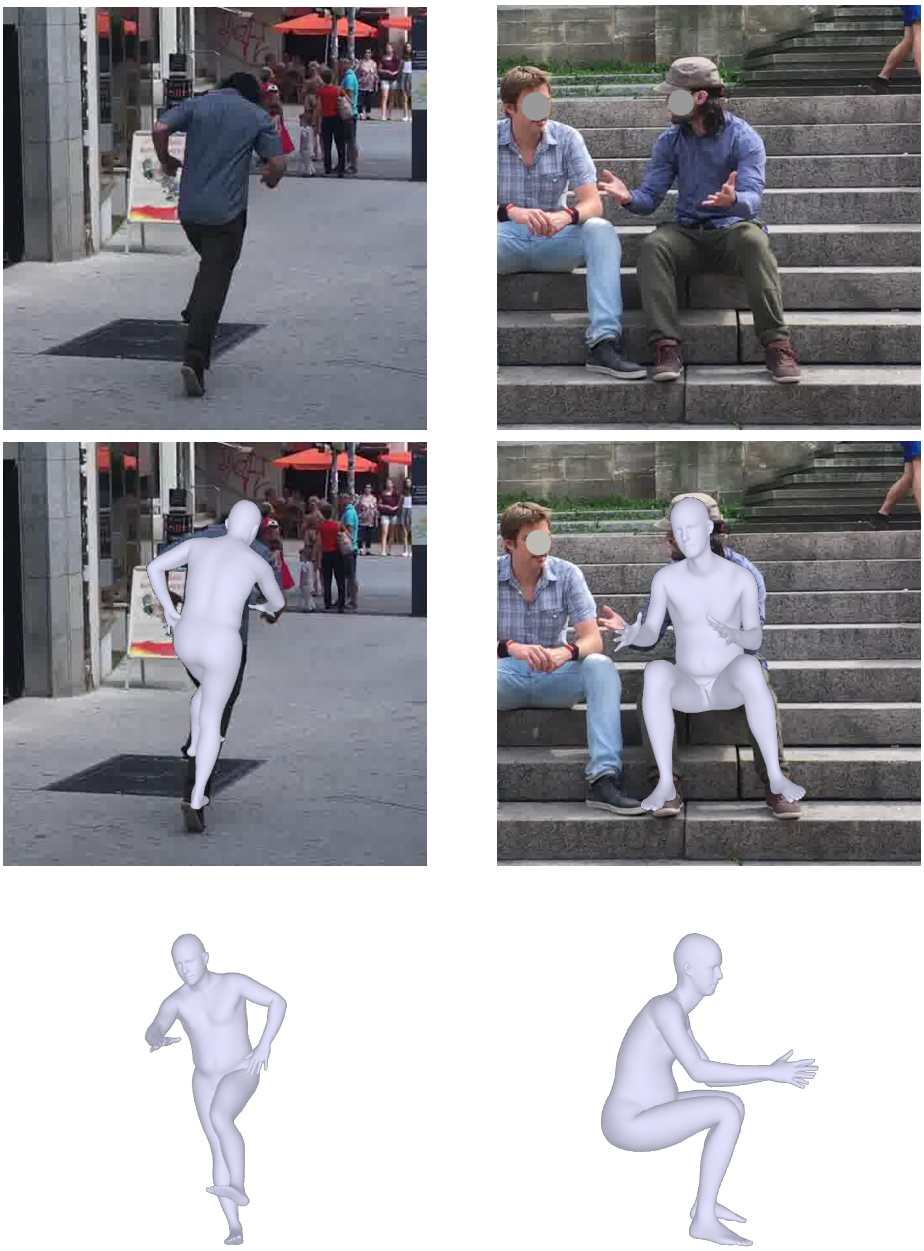


Figure 5.1. Example solutions from our motion capture framework based on our proposed sparse constrained optimization. (left) input image from the 3DPW [**10**] dataset, (middle) 3D pose and shape reconstruction overlayed on the input image, (right) 3D reconstruction shown from a rotated viewpoint.

constrained formulation that is mathematically equivalent under mild assumptions. We show how the underlying sparsity and constraints of our formulation can be exploited leading to sparse Hessian matrices and ultimately computing the Gauss-Newton direction in $O(K) + O(N)$ time for a 3D human model with K joints and N measurements. Our optimization method, *SCOPE (Sparse Constrained Optimization for 3D human Pose and shapE estimation)*, is thus orders of magnitude faster (average 4 ms convergence) than existing optimization methods, particularly when the number of joints K and measurements N is large.

Based on our optimization method, we present a real-time 3D motion capture framework (illustrated in Figure 5.2) that estimates 3D human poses and shapes from a single image at over 30 FPS. Example solutions are shown in Figure 5.1. Our method allows using a modified SMPL model [**144**] that has 75 degrees of freedom and 10 shape parameters, and estimates both human poses and shapes with which the 3D human mesh can be reconstructed. In contrast, several real-time 3D motion capture frameworks using optimization methods [**137**, **138**] adopt a much simpler 3D skeleton model with 33 degrees of freedom and no shape parameters to reduce the computation complexity and are therefore unable to reconstruct the 3D human mesh. We compare our real-time 3D motion capture framework with numerous state-of-the-art methods [**2**, **11**, **12**, **139**, **141**, **145**] on public benchmark datasets [**10**, **13**, **14**]. Our framework achieves accuracies that outperform optimization methods [**11**, **12**, **138**, **139**] and are competitive to regression methods [**2**, **141**].

In summary, our contributions are: (i) we propose a sparse constrained formulation for 3D human pose and shape estimation that is mathematically equivalent to the dense unconstrained formulation of existing optimization methods under mild assumptions; (ii)

we develop an efficient algorithm that computes the Gauss-Newton direction in linear-time complexity with respect to the number of joints and measurements; and (iii) we present a real-time 3D motion capture framework that estimates 3D human poses and shapes from a single image.

The rest of this chapter is organized as follows. Section 5.2 reviews the state-of-the-art optimization and regression methods for 3D human pose and shape estimation. Section 5.3 formulates the optimization problem. Section 5.4 proposes the sparse constrained formulation as well as the $O(K)$ algorithm to compute the Gauss-Newton direction. Section 5.5 presents the real-time motion capture framework for 3D human pose and shape estimation. Sections 5.6 and 5.7 evaluate the proposed method on various benchmark datasets and make comparisons against existing state-of-the-art methods. Section 5.8 concludes this chapter and discusses future work. Section 5.9 proves the propositions presented in this chapter and presents a complete complexity analysis.

5.2. Related work

Optimization methods estimate human poses and shapes by matching 3D joints on the human body to 2D keypoints on the image. Works in human body modeling [**144**, **146**, **147**] and 2D keypoint detection [**148**–**150**] have made substantial contributions, but the resulting optimization problem remains challenging due to the ambiguity in the 3D information from an image and the uncertainty of 3D human poses. To address this, recent works have incorporated 3D information, such as 3D keypoint positions [**137**,**138**], part orientation fields [**11**], silhouette [**151**], etc, as additional fitting

terms. Additionally, human 3D pose priors in the form of mixture of Gaussians [**12**], variational auto-encoder [**152**], and normalizing flow [**153**] have been trained from numerous datasets [**13**, **154**, **155**] and successfully applied to human 3D pose and shape estimation. A closer look at these optimization methods [**11**, **12**, **137**–**139**, **153**] does reveal that they primarily differ in their loss terms of the objective function while still utilizing the same underlying dense unconstrained formulation. We show that such a formulation is inherently inefficient in computing the Gauss-Newton direction. Thus despite the considerable progress, these methods still take tens of seconds to converge and are impractical for real time applications.

Regression methods use deep neural networks to regress human poses and shapes directly from images. In most cases, regression methods [**2**, **141**, **145**, **156**] take only tens of milliseconds to process one image and meet the real-time requirements. Unlike [**157**–**161**] that lift 2D keypoints to 3D keypoints, regression methods for 3D human pose and shape estimation face a challenge in having access to large datasets with ground truth labels of 3D human pose and shape. To address this, regressions methods often employ optimization methods to precompute 3D ground truth for supervision [**141**] or even have optimization methods in the loop [**2**] during training. Other examples like [**156**] rely on optimization methods to refine the network outputs. In these aforementioned scenarios, the computational efficiency of optimization methods play an important role both during training and deployment.

5.3. Problem Formulation

5.3.1. SMPL Model

The SMPL model [**144**] is a vertex-based linear blend skinning 3D human model. In this chapter, we use a SMPL model that has $K = 23$ rotational joints, $V = 6890$ vertices, and $P = 10$ shape parameters.

The SMPL model represents the human body using a kinematic tree with $K+1$ interconnected body parts indexed with $i = 0, 1, \cdots, K$. In the rest of this chapter, we use $\text{par}(i)$ to denote the parent of body part i, and $\mathbf{T}_i \in SE(3)$ the pose of body part i, and $\boldsymbol{\Omega}_i \in SO(3)$ the state of joint i, and $\boldsymbol{\beta} \in \mathbb{R}^P$ the shape parameters. Note that body part i is connected to its parent body part $\text{par}(i)$ through joint i.

Let $\mathbf{T}_i \triangleq \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$ denote the pose of body part i where $\mathbf{R}_i \in SO(3)$ is the rotation and $\mathbf{t}_i \in \mathbb{R}^3$ is the translation as well as the position of joint i. The SMPL model [**144**] assumes that the joint positions at the rest pose linearly depend on the vertex positions, and the vertex positions at the rest pose linearly depend on the shape parameters $\boldsymbol{\beta} \in \mathbb{R}^P$. Thus, we conclude that the joint positions $\bar{\mathbf{t}}_i \in \mathbb{R}^3$ at the rest pose linearly depend on the shape parameters $\boldsymbol{\beta}$, i.e., there exists $\mathcal{J}_i \in \mathbb{R}^{3\times P}$ and $\mathbf{c}_i \in \mathbb{R}^3$ in the SMPL model such that $\bar{\mathbf{t}}_i$ at the rest pose takes the form of

$$\bar{\mathbf{t}}_i = \mathcal{J}_i \cdot \boldsymbol{\beta} + \mathbf{c}_i. \tag{5.1}$$

Moreover, the relative joint position $\Delta\bar{\mathbf{t}}_i \in \mathbb{R}^3$ between any connected body parts is constant, and thus, we obtain $\Delta\bar{\mathbf{t}}_i = \bar{\mathbf{t}}_i - \bar{\mathbf{t}}_{\text{par}(i)}$, where $\text{par}(i)$ denotes the index of the parent of body part i. Then, joint position $\mathbf{t}_i \in \mathbb{R}^3$ at any poses satisfies

$$\mathbf{t}_i = \mathbf{R}_{\text{par}(i)} \Delta\overline{\mathbf{t}} + \mathbf{t}_{\text{par}(i)} = \mathbf{R}_{\text{par}(i)} \big(\overline{\mathbf{t}}_i - \overline{\mathbf{t}}_{\text{par}(i)}\big) + \mathbf{t}_{\text{par}(i)}. \tag{5.2}$$

In the equation above, $\mathbf{R}_{\text{par}(i)}$ is rotation of pose $\mathbf{T}_{\text{par}(i)} \in SE(3)$. Substituting Eq. (5.1) into Eq. (5.2) to cancel out $\overline{\mathbf{t}}_i$ and $\overline{\mathbf{t}}_{\text{par}(i)}$, we obtain

$$\mathbf{t}_i = \mathbf{R}_{\text{par}(i)} \big(\mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i\big) + \mathbf{t}_{\text{par}(i)}, \tag{5.3}$$

where

$$\mathcal{S}_i = \mathcal{J}_i - \mathcal{J}_{\text{par}(i)} \in \mathbb{R}^{3\times P} \tag{5.4}$$

and

$$\mathbf{l}_i = \mathbf{c}_i - \mathbf{c}_{\text{par}(i)} \in \mathbb{R}^3. \tag{5.5}$$

It is immediate to show that $\mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i$ is the relative joint position between body parts i and par(i), and thus, the corresponding relative pose $\mathbf{T}_{\text{par}(i),i}$ is

$$\mathbf{T}_{\text{par}(i),i} \triangleq \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.6}$$

where $\boldsymbol{\Omega}_i \in SO(3)$ is the state of joint i. Then $\mathbf{T}_i$ can be recursively computed as

$$\mathbf{T}_i = \mathbf{T}_{\text{par}(i)} \mathbf{T}_{\text{par}(i),i} = \mathbf{T}_{\text{par}(i)} \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5.7}$$

5.3.2. Rigid Skinning Assumption of Keypoints

We need to select a set of joints and vertices on the SMPL model as keypoints to calculate 2D and 3D keypoint losses, part orientation field losses, etc. [**11**, **12**, **137**, **138**].

In this chapter, we modify the SMPL model and make the following assumption of the selected keypoints for loss calculation.

Assumption 5.1. Each keypoint j is rigidly attached to a body part i, i.e., the position $\mathbf{v}_j \in \mathbb{R}^3$ of keypoint j is

$$\mathbf{v}_j = \mathbf{R}_i \overline{\mathbf{v}}_j + \mathbf{t}_i, \tag{5.8}$$

where $\mathbf{R}_i \in SO(3)$ and $\mathbf{t}_i \in \mathbb{R}^3$ are the rotation and translation of pose $\mathbf{T}_i \in SE(3)$, and $\overline{\mathbf{v}}_j \in \mathbb{R}^3$ is the relative position of keypoint j with respect to body part i. Furthermore, there exists $\mathcal{V}_j \in \mathbb{R}^{3\times P}$ and $\overline{\mathbf{v}}_{j,0} \in \mathbb{R}^3$ such that the relative position $\overline{\mathbf{v}}_j \in \mathbb{R}^3$ in Eq. (5.8) is evaluated as

$$\overline{\mathbf{v}}_j = \mathcal{V}_j \cdot \boldsymbol{\beta} + \overline{\mathbf{v}}_{j,0}. \tag{5.9}$$

For simplicity, we use $\mathcal{V}_j$ and $\overline{\mathbf{v}}_{j,0}$ extracted from the joint and vertex positions at the rest pose of the SMPL model, whose derivation is similar to that of $\mathcal{S}_i$ and $\mathbf{l}_i$ in Eq. (5.6). We remark that Assumption 5.1 is important for our sparse constrained formulation presented later in this chapter.

Compared to the SMPL model, Assumption 5.1 keeps rigid skinning (shape blend shapes) while dropping nonrigid skinning (pose blend shapes) for the vertex keypoints. We argue that Assumption 5.1 is a reasonable and mild modification for human pose and shape estimation. First, the SMPL model evaluates the joint keypoints, such as wrists, elbows, knees, etc, using Eq. (5.7), which is essentially equivalent to Eqs. (5.8) and (5.9) of rigid skinning. While the SMPL model has each vertex position depend on the poses of all the body parts, the vertices selected as keypoints, such as nose, eyes, ears, etc., are

mainly affected by a single body part. Finally, we note that inaccuracies are also present in 2D and 3D keypoint measurements used for estimation, which are usually much larger than those induced by the SMPL model modification using Eqs. (5.8) and (5.9).

5.3.3. Objective Function

Given an RGB image, we use the following objective for human pose and shape estimation:

$$\mathtt{E} = \sum_{0 \leq i \leq K} \left(\mathtt{E}_{\mathrm{2D},i} + \lambda_{\mathrm{3D}} \cdot \mathtt{E}_{\mathrm{3D},i} + \lambda_{\mathrm{p}} \cdot \mathtt{E}_{\mathrm{p},i} + \lambda_{\mathrm{T}} \cdot \mathtt{E}_{\mathrm{T},i} + \lambda_{\Omega} \cdot \mathtt{E}_{\Omega,i} \right) + \lambda_{\beta} \cdot \mathtt{E}_{\beta}, \tag{5.10}$$

where λ_{3D}, λ_{p}, λ_{T}, λ_{Ω} and λ_{β} are scalar weights and joint state $\mathbf{\Omega}_0 \in SO(3)$ for body part 0 is a dummy variable. Each loss term in Eq. (5.10) is defined as follows:

(1) $\mathtt{E}_{\mathrm{2D},i} \triangleq \frac{1}{2} \sum_{j \in \mathrm{V}_{\mathrm{2D},i}} \| \Pi_{\mathbf{K}}(\mathbf{v}_j) - \hat{\mathbf{v}}_{\mathrm{2D},j} \|^2$ is the 2D keypoint loss, where $\mathrm{V}_{\mathrm{2D},i}$ is the set of indices of keypoints attached to body part i and selected to calculate the 2D keypoint loss, $\Pi_{\mathbf{K}}(\cdot)$ is the 3D to 2D projection map with camera intrinsics $\mathbf{K}$, $\mathbf{v}_j \in \mathbb{R}^3$ is the keypoint position, and $\hat{\mathbf{v}}_{\mathrm{2D},j} \in \mathbb{R}^2$ is the 2D keypoint measurement.

(2) $\mathtt{E}_{\mathrm{3D},i} \triangleq \frac{1}{2} \sum_{j \in \mathrm{V}_{\mathrm{3D},i}} \| \mathbf{v}_j - \hat{\mathbf{v}}_{\mathrm{3D},j} \|^2$ is the 3D keypoint loss, where $\mathrm{V}_{\mathrm{3D},i}$ is the set of indices of keypoints attached to body part i and selected to calculate the 3D keypoint loss, $\mathbf{v}_j \in \mathbb{R}^3$ is the keypoint position and $\hat{\mathbf{v}}_{\mathrm{3D},j} \in \mathbb{R}^3$ is the 3D keypoint measurement.

(3) $\mathtt{E}_{\mathrm{p},i} \triangleq \frac{1}{2} \sum_{j \in \mathrm{P}_i} \left\| \frac{\mathbf{v}_j - \mathbf{t}_i}{\| \mathbf{v}_j - \mathbf{t}_i \|} - \hat{\mathbf{p}}_j \right\|^2$ is the part orientation field loss [**11**], where P_i is the set of indices of keypoints attached to body part i and selected to calculate the part orientation field loss, $\mathbf{v}_j \in \mathbb{R}^3$ is the keypoint position, and $\mathbf{t}_i \in \mathbb{R}^3$ is the position of body part i as well as the translation of pose $\mathbf{T}_i \in SE(3)$, and $\hat{\mathbf{p}}_i \in \mathbb{R}^3$ is the part orientation field measurement.

(4) $\mathtt{E}_{\mathrm{T},i} \triangleq \frac{1}{2}\|\mathbf{T}_i - \hat{\mathbf{T}}_i\|^2$ is the prior loss of pose $\mathbf{T}_i \in SE(3)$, where $\hat{\mathbf{T}}_i \in SE(3)$ is a known prior estimate.

(5) $\mathtt{E}_{\Omega,i} \triangleq \frac{1}{2}\|\mathtt{r}_{\Omega_i}(\boldsymbol{\Omega}_i)\|^2$ is the prior loss of joint state $\boldsymbol{\Omega}_i \in SO(3)$, where $\mathtt{r}_{\Omega_i}(\cdot)$ is a normalizing flow of $SO(3)$ trained on the AMASS dataset **[154]**.

From the definitions above, each loss term $\mathtt{E}_{(\#),i}$ in Eq. (5.10) can be in general formulated as

$$\mathtt{E}_{(\#),i} = \sum_j \frac{1}{2}\|\mathtt{r}_{(\#),ij}(\mathbf{T}_i,\, \boldsymbol{\Omega}_i,\, \boldsymbol{\beta},\, \mathbf{v}_j)\|^2, \tag{5.11}$$

where $\mathtt{r}_{(\#),ij}(\cdot)$ is a function of $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$, $\boldsymbol{\beta}$ and $\mathbf{v}_j$. Since keypoint j in Eq. (5.11) is attached to body part i, then Eqs. (5.8) and (5.9) indicate that $\mathbf{v}_j$ is a function of $\mathbf{T}_i$ and $\boldsymbol{\beta}$:

$$\mathbf{v}_j = \mathbf{R}_i\big(\mathcal{V}_j \cdot \boldsymbol{\beta} + \overline{\mathbf{v}}_{j,0}\big) + \mathbf{t}_i. \tag{5.12}$$

As a result of Eq. (5.12), we can cancel out $\mathbf{v}_j$ in Eq. (5.11) and simplify $\mathtt{r}_{(\#),ij}(\cdot)$ as a function of $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$ and $\boldsymbol{\beta}$:

$$\mathtt{E}_{(\#),i} = \sum_j \frac{1}{2}\|\mathtt{r}_{(\#),ij}(\mathbf{T}_i,\, \boldsymbol{\Omega}_i,\, \boldsymbol{\beta})\|^2. \tag{5.13}$$

We remark that $\mathtt{r}_{(\#),ij}(\cdot)$ in Eq. (5.13) is related to $\mathbf{T}_i \in SE(3)$ and $\boldsymbol{\Omega}_i \in SO(3)$ of a single body part i. Then, Eq. (5.13) immediately suggests that Eq. (5.10) takes the form of

$$\mathtt{E} = \sum_{0\le i\le K} \frac{1}{2}\|\mathtt{r}_i(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta})\|^2, \tag{5.14}$$

where each $\mathtt{r}_i(\cdot)$ is a function of $\mathbf{T}_i \in SE(3)$, $\boldsymbol{\Omega}_i \in SO(3)$ and $\boldsymbol{\beta} \in \mathbb{R}^P$. Besides those in Eq. (5.10), a number of losses can be written in the form of Eqs. (5.11) and (5.13) as well.

5.3.4. Dense Unconstrained Optimization

With Eqs. (5.6) and (5.7), we might recursively compute each $\mathbf{T}_i \in SE(3)$ through a top-down traversal of the kinematics tree. Thus, each $\mathbf{T}_i$ can be written as a function of the root pose $\mathbf{T}_0 \in SE(3)$, the joint states $\boldsymbol{\Omega} \triangleq (\boldsymbol{\Omega}_0, \boldsymbol{\Omega}_1, \cdots, \boldsymbol{\Omega}_K) \in SO(3)^{K+1}$ and the shape parameters $\boldsymbol{\beta} \in \mathbb{R}^P$:

$$\mathbf{T}_i \triangleq \mathbf{T}_i\left(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}\right). \tag{5.15}$$

In existing optimization methods [**11**, **12**, **137**–**139**, **152**], Eq. (5.15) is substituted into Eq. (5.14) to cancel out non-root poses $\mathbf{T}_i \in SE(3)$ $(1 \leq i \leq K)$, which results in a dense unconstrained optimization problem of $\mathbf{T}_0 \in SE(3)$, $\boldsymbol{\Omega} \in SO(3)^K$ and $\boldsymbol{\beta} \in \mathbb{R}^P$:

$$\min_{\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}} \mathtt{E} = \sum_{0 \leq i \leq K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta})\|^2. \tag{5.16}$$

In general, Gauss-Newton is the preferred method to solve optimization problems of the kind in Eq. (5.16). This consists of linearization to find the Jacobian matrix, building and then solving the linear system to find the Gauss-Newton direction. In Section 5.9.2, we show that Eq. (5.16) yields a dense linear system when computing the Gauss-Newton direction. Since the complexity of dense linear system computation increases superlinearly with their size, the dense unconstrained formulation of Eq. (5.16) has poor scalability when the human model has large numbers of joints and measurements.

5.4. Method

In this section, we present a sparse constrained formulation for 3D human pose and shape estimation that is mathematically equivalent to the dense unconstrained one in

Section 5.3.4. From our formulation, we derive a method that scales linearly with the number of joints and measurements to compute the Gauss-Newton direction.

5.4.1. Sparse Constrained Optimization

We introduce $\boldsymbol{\beta}_i \in \mathbb{R}^P$ with $\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\text{par}(i)}$ for each body part i in the SMPL model. Since $\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\text{par}(i)}$ indicates $\boldsymbol{\beta}_i = \boldsymbol{\beta}$, and $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$ and $\boldsymbol{\beta}$ satisfy the kinematic constraints of Eq. (5.7), we formulate 3D human pose and shape estimation of Eq. (5.14) as a sparse constrained optimization problem on $\{\mathbf{T}_i, \boldsymbol{\beta}_i, \boldsymbol{\Omega}_i\}_{i=0}^K \in \big(SE(3) \times \mathbb{R}^P \times SO(3)\big)^{K+1}$:

$$\min_{\{\mathbf{T}_i, \boldsymbol{\beta}_i, \boldsymbol{\Omega}_i\}_{i=0}^K} \sum_{0 \le i \le K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta}_i)\|^2 \tag{5.17}$$

subject to

$$\begin{aligned} \mathbf{T}_i &= \mathtt{F}_i(\mathbf{T}_{\text{par}(i)}, \boldsymbol{\beta}_{\text{par}(i)}, \boldsymbol{\Omega}_i) \\ &\triangleq \mathbf{T}_{\text{par}(i)} \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta}_{\text{par}(i)} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{5.18a}$$

$$\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\text{par}(i)}. \tag{5.18b}$$

In Eq. (5.18a), note that $\mathtt{F}_i(\cdot) : SE(3) \times \mathbb{R}^P \times SO(3) \to SE(3)$ is a function corresponding to Eq. (5.7) and maps $\mathbf{T}_{\text{par}(i)}$, $\boldsymbol{\beta}_{\text{par}(i)}$, $\boldsymbol{\Omega}_i$ to $\mathbf{T}_i$. For notational simplicity, we define $\mathbf{x}_i \triangleq (\mathbf{T}_i, \boldsymbol{\beta}_i) \in SE(3) \times \mathbb{R}^P$. Then, Eqs. (5.17) and (5.18) are equivalent to

$$\min_{\{\mathbf{x}_i, \boldsymbol{\Omega}_i\}_{i=0}^K} \sum_{0 \le i \le K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{x}_i, \boldsymbol{\Omega}_i)\|^2 \tag{5.19}$$

subject to

$$\mathbf{x}_i = \begin{bmatrix} \mathtt{F}_i(\mathbf{x}_{\mathrm{par}(i)},\, \boldsymbol{\Omega}_i) \\ \\ \boldsymbol{\beta}_{\mathrm{par}(i)} \end{bmatrix}. \tag{5.20}$$

In spite of additional optimization variables and kinematic constraints compared to Eq. (5.16), we have the following proposition for our sparse constrained formulation.

Proposition 5.4.1. Eqs. (5.19) and (5.20) are equivalent to Eq. (5.16) under Assumption 5.1.

Proof. Please refer to Section 5.9.1. □

In the remainder of this section, we will make use of the sparsity and constraints of Eqs. (5.19) and (5.20) to simplify the computation of the Gauss-Newton direction.

5.4.2. Gauss-Newton Direction

The computation of the Gauss-Newton direction for Eqs. (5.19) and (5.20) is summarized as follows.

Step 1: The linearization of Eqs. (5.19) and (5.20) results in

$$\min_{\{\Delta\mathbf{x}_i,\, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}} \Delta\mathtt{E} = \sum_{0\leq i\leq K} \frac{1}{2}\left\|\mathbf{J}_{i,1}\Delta\mathbf{x}_i + \mathbf{J}_{i,2}\Delta\boldsymbol{\Omega}_i + \mathtt{r}_i\right\|^2, \tag{5.21}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i, \tag{5.22}$$

where $\Delta\mathbf{x}_i \triangleq (\Delta\mathbf{T}_i,\, \Delta\boldsymbol{\beta}_i) \in \mathbb{R}^{6+P}$ and $\Delta\boldsymbol{\Omega}_i \in \mathbb{R}^3$ are the Gauss-Newton directions of $\mathbf{x}_i$ and $\boldsymbol{\Omega}_i$, respectively, and $\mathtt{r}_i$ in Eq. (5.21) is the residue, and

$$\mathbf{J}_{i,1} \triangleq \frac{\partial \mathtt{r}_i}{\partial \mathbf{x}_i} = \begin{bmatrix} \dfrac{\partial \mathtt{r}_i}{\partial \mathbf{T}_i} & \dfrac{\partial \mathtt{r}_i}{\boldsymbol{\beta}_i} \end{bmatrix} \text{ and } \mathbf{J}_{i,2} \triangleq \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\Omega}_i}, \tag{5.23}$$

in Eq. (5.21) are the Jacobians, and

$$\mathbf{A}_i \triangleq \begin{bmatrix} \dfrac{\partial \mathtt{F}_i}{\partial \mathbf{T}_{\mathrm{par}(i)}} & \dfrac{\partial \mathtt{F}_i}{\partial \boldsymbol{\beta}_{\mathrm{par}(i)}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \text{ and } \mathbf{B}_i \triangleq \begin{bmatrix} \dfrac{\partial \mathtt{F}_i}{\partial \boldsymbol{\Omega}_i} \\ \mathbf{0} \end{bmatrix} \tag{5.24}$$

in Eq. (5.22) are the partial derivatives of Eq. (5.20). For $\Delta\mathbf{x}_i = (\Delta\mathbf{T}_i,\, \Delta\boldsymbol{\beta}_i) \in \mathbb{R}^{6+P}$ in Eqs. (5.21) and (5.22), note that $\Delta\mathbf{T}_i \in \mathbb{R}^6$ and $\Delta\boldsymbol{\beta}_i \in \mathbb{R}^P$ are the Gauss-Newton direction of $\mathbf{T}_i$ and $\boldsymbol{\beta}_i$, respectively.

Step 2: We reformulate Eqs. (5.21) and (5.22) as

$$\min_{\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^K} \Delta\mathtt{E} = \sum_{i=0}^{K} \Big[\frac{1}{2} \Delta\mathbf{x}_i^\top \mathbf{H}_{i,11} \Delta\mathbf{x}_i + \Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,21} \Delta\mathbf{x}_i + \frac{1}{2} \Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,22} \Delta\boldsymbol{\Omega}_i + \mathbf{g}_{i,1}^\top \Delta\mathbf{x}_i + \mathbf{g}_{i,2}^\top \Delta\boldsymbol{\Omega}_i \Big], \tag{5.25}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i \Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i \Delta\boldsymbol{\Omega}_i, \tag{5.26}$$

where $\mathbf{H}_{i,11} \triangleq \mathbf{J}_{i,1}^\top \mathbf{J}_{i,1}$, $\mathbf{H}_{i,21} \triangleq \mathbf{J}_{i,2}^\top \mathbf{J}_{i,1}$ and $\mathbf{H}_{i,22} \triangleq \mathbf{J}_{i,2}^\top \mathbf{J}_{i,2}$ are the Hessians, and $\mathbf{g}_{i,1} \triangleq \mathbf{J}_{i,1}^\top \mathtt{r}_i$ and $\mathbf{g}_{i,2} \triangleq \mathbf{J}_{i,2}^\top \mathtt{r}_i$ are the gradients.

Step 3: Implement Algorithm 15 to solve Eqs. (5.25) and (5.26) and compute the Gauss-Newton direction $\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^K$.

Here, **Steps 1 to 3** compute the Gauss-Newton direction $\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}$ by solving a constrained quadratic optimization problem. The following proposition is for its completeness and complexity.

Proposition 5.4.2. The resulting $\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}$ for Eqs. (5.19) and (5.20) is also the Gauss-Newton direction for Eq. (5.16). Furthermore, Eqs. (5.19) and (5.20) take $O(K) + O(N)$ time to compute $\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}$ using **Steps 1 to 3**, where K and N are the number of joints and measurements of the 3D human model, respectively. In contrast, Eq. (5.16) has a complexity of $O(K^3) + O(K^2N)$.

Proof. Please refer to Section 5.9.2. □

In general, the computation of the Gauss-Newton direction occupies a significant portion of workloads in optimization. Since our sparse constrained formulation improves this computation by two orders in terms of the number of joints and has the number of joints and measurements decoupled for the complexity, it is expected that our resulting method greatly improves the efficiency of optimization.

5.5. Real-time Motion Capture Framework

We design a real-time monocular motion capture framework, illustrated in Figure 5.2, based on our fast optimization method to recover 3D human poses and shapes from a single image. Similar to the other frameworks [**137**, **138**], ours consists of a preprocessing pipeline with the input image fed to YOLOv4-CSP [**3**, **162**] for human detection, then to AlphaPose [**149**] for 2D keypoint estimation, and finally to a light-weight neural network that is a modification of VideoPose3D [**158**] for 2D-to-3D lifting. The output of the

Algorithm 15 Solve Eqs. (5.25) and (5.26) and compute the Gauss-Newton direction

Input:$\{\mathbf{H}_{i,11}, \mathbf{H}_{i,21}, \mathbf{H}_{i,22}, \mathbf{g}_{i,1}, \mathbf{g}_{i,2}\}_{i=0}^{K}$
Output:$\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}$ and $\Delta\mathtt{E}_0$
for $i = K \to 1$ **do**
    $\mathbf{N}_{i,11} = \mathbf{H}_{i,11} + \sum_{j\in\mathrm{chd}(i)} \mathbf{M}_{j,11}$
    $\mathbf{N}_{i,21} = \mathbf{H}_{i,21}$
    $\mathbf{N}_{i,22} = \mathbf{H}_{i,22}$
    $\mathbf{n}_{1,i} = \mathbf{g}_{i,1} + \sum_{j\in\mathrm{chd}(i)} \mathbf{m}_{j,1}$
    $\mathbf{n}_{i,2} = \mathbf{g}_{i,2}$
    $\Delta\overline{\mathtt{E}}_i = \sum_{j\in\mathrm{chd}(i)} \Delta\mathtt{E}_j$
    $\mathbf{Q}_{i,11} = \mathbf{A}_i^\top \mathbf{N}_{i,11}\mathbf{A}_i$
    $\mathbf{Q}_{i,21} = \mathbf{B}_i^\top \mathbf{N}_{i,11}\mathbf{A}_i + \mathbf{N}_{i,21}\mathbf{A}_i$
    $\mathbf{Q}_{i,22} = \mathbf{B}_i^\top \mathbf{N}_{i,11}\mathbf{B}_i + \mathbf{N}_{i,21}\mathbf{B}_i + \mathbf{B}_i^\top \mathbf{N}_{i,21}^\top + \mathbf{N}_{i,22}$
    $\mathbf{q}_{i,1} = \mathbf{A}_i^\top \mathbf{n}_{i,1}$
    $\mathbf{q}_{i,2} = \mathbf{B}_i^\top \mathbf{n}_{i,1} + \mathbf{n}_{i,2}$
    $\mathbf{K}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21}$
    $\mathbf{k}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}$
    $\mathbf{M}_{i,11} = \mathbf{Q}_{i,11} - \mathbf{Q}_{i,21}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21}$
    $\mathbf{m}_{1,i} = \mathbf{q}_{i,1} - \mathbf{Q}_{i,21}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}$
    $\Delta\mathtt{E}_i = \Delta\overline{\mathtt{E}}_i - \frac{1}{2}\mathbf{q}_{i,2}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}$
end for
$\Delta\boldsymbol{\Omega}_0 = \mathbf{0}$
$\mathbf{M}_0 = \mathbf{H}_{0,11} + \sum_{j\in\mathrm{chd}(0)} \mathbf{M}_{j,11}$
$\mathbf{m}_0 = \mathbf{g}_{0,1} + \sum_{j\in\mathrm{chd}(0)} \mathbf{m}_{j,1}$
$\Delta\overline{\mathtt{E}}_0 = \sum_{i\in\mathrm{chd}(0)} \Delta\mathtt{E}_i$
$\mathbf{x}_0 = -\mathbf{M}_0^{-1}\mathbf{m}_0$
$\Delta\mathtt{E}_0 = \Delta\overline{\mathtt{E}}_0 - \frac{1}{2}\mathbf{m}_0^\top \mathbf{M}_0^{-1}\mathbf{m}_0$
for $i = 1 \to K$ **do**
    $\Delta\boldsymbol{\Omega}_i = \mathbf{K}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{k}_i$
    $\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i$
end for

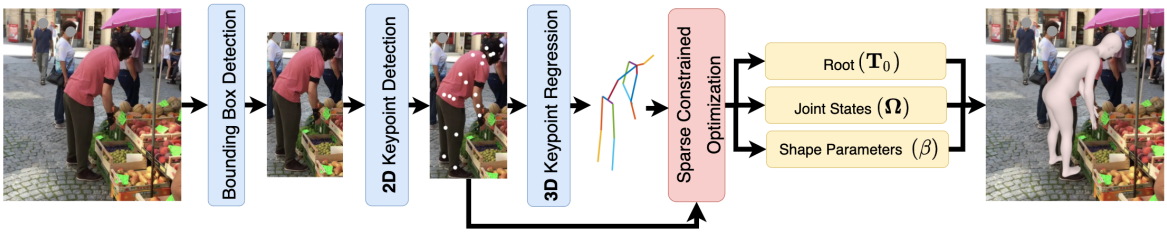


Figure 5.2. Overview of our motion capture framework. Given an image, our preprocessing pipeline estimates a bounding box, 2D and 3D keypoints. The 2D and 3D keypoints are then sent to our fast sparse constrained optimizer for 3D pose and shape reconstruction. Note that 3D keypoints are used to compute the part orientation fields [**11**].

preprocessing pipeline is then sent to our fast optimizer for 3D reconstruction. The Python API of NVIDIA TensorRT 7.2.1 is used to accelerate the inference of the preprocessing neural networks.

5.5.1. Human Detection

The YOLOv4-CSP [**3**, **162**] is used for human detection to balance between accuracy and efficiency. The size of input images for YOLOv4-CSP is 512×512.

5.5.2. 2D Keypoint Estimation

The AlphaPose [**149**] is used for 2D keypoint estimation with 256×192 input images. The following datasets are used to train AlphaPose.

Human3.6M [**13**] is a popular dataset for 3D human pose estimation. Following the standard training-testing protocol in [**157**], we use subjects S1, S5-S8 for training.

MPI-INF-3DHP [**14**] is a multi-view markerless dataset with 8 training subjects and 6 test subjects. We use subjects S1-S8 that are downsampled to 10 FPS for training.

COCO [**163**] is a large-scale dataset for 2D joint detection. We use the COCO training datasets for training.

MPII [**164**] is a 2D human pose dataset that is extracted from online videos. We use the MPII training datasets for training.

5.5.3. 3D Keypoint Regression

In our real-time motion capture framework, we use a light-weight fully connected neural network for 2D-to-3D lifting. The 3D Keypoint regression network is a modification of VideoPose3D [**158**]. From the 3D keypoint regression network, we further obtain the part orientation field [**11**] for each body part. We use the training datasets of Human3.6M [**13**] and MPI-INF-3DHP [**14**] that are downsampled to 10 FPS to train the 3D keypoint regression network.

5.6. Evaluation

In this section, we present quantitative and qualitative evaluation of our method against state-of-the-art optimization and regression methods on multiple public benchmark datasets. All experiments are done on an Intel Xeon E3-1505M 3.0GHz CPU and a NVIDIA Quadro GP 100 GPU.

5.6.1. Datasets

We evaluate all methods on the following datasets.

Human3.6M (H36M) [**13**] is one of the most commonly used datasets for 3D human pose (and shape) estimation (it was obtained and used by coauthors affiliated with academic institutions). Following the standard training-testing protocol established in [**157**], we use subjects S9 and S11 for evaluation.

MPI-INF-3DHP [**14**] is a markerless dataset with multiple viewpoints. We use subjects TS1-TS6 for evaluation where the first four (TS1-TS4) are in a controlled lab environment and the last two are in the wild (TS5-TS6).

3DPW [**10**] is an in-the-wild dataset captured from a moving single hand-held camera. IMU sensors are also used to compute ground-truth poses and shapes using the SMPL model. We use its defined test dataset for evaluation.

5.6.2. Computation Times

We evaluate all methods on their computation or inference times on the Human3.6M dataset [**13**] dataset. We compare optimization methods against ours on the optimization only time and compare all methods on the total computation time per image.

Optimization time is reported in column 4 of Table 5.1. Our method converges in 20-50 iterations taking less than 4ms on average to reconstruct 3D human poses and shapes. In contrast to existing optimization methods that estimate pose and shape [**11**, **12**, **139**] in 20-45s, ours is 4 orders of magnitude faster. As discussed earlier, our method uses the SPML model with 2.6 times as many variables (75 degrees of freedom and 10 shape parameters) as the 3D skeleton in VNect [**138**] (33 degrees of freedom and no shape parameters)—note that the complexity of optimization problems typically increases superlinearly with the number of optimization variables. Our optimization method is still

twice as fast as VNect that only estimates poses (with an objective function with fewer loss terms). We attribute the significant improvements in optimization times to our sparse constrained formulation whose computation of the Gauss-Newton direction has linear rather than cubic complexity with the number of joints and measurements. The ablation studies in Section 5.7 further support our complexity analysis.

Total time includes the preprocessing time and any optimization or regression time and reflects the overall time it takes for a method to produce estimates given an image. All timings are reported in columns 3-6 of Table 5.1. The regression methods [**2**, **141**, **145**] use ground-truth bounding boxes during evaluation. Therefore, we assume YOLOv4-CSP [**3**, **162**] (17ms) is used in practice to obtain bounding boxes from images and count it as the preprocessing time per image. For the optimization methods, the preprocessing time of VNect [**138**] is computed from its own neural networks while for others [**11**,**12**,**139**] the preprocessing pipeline is similar to ours and we assume their times (29ms) are close to ours. Note that in our method the 29ms preprocessing time is a significant portion of the total time, while for the other optimization methods (that estimate pose and shape) it is negligible compared to their optimization times. SPIN [**2**] has the lowest total time of 29ms and ours is a close second with 33ms. Our motion capture framework thus has a speed of over 30 FPS which is sufficient for real-time applications.

5.6.3. Accuracy

Human3.6M. We evaluate all methods on the Mean Per-Joint Position Errors without (MPJPE) and with (PA-MPJPE) Procrustes Alignment on two common protocols. Protocol 1 uses all the four cameras and Protocol 2 only uses the frontal camera. The

results are reported in columns 7-9 of Table 5.1. Our framework outperforms the other methods on Protocol 1 MPJPE, and achieves the second lowest PA-MPJPE slightly behind SPIN [**2**] on both Protocols 1 and 2. Though not presented in Table 5.1, our method also has the lowest MPJPE on Protocol 2, which is 60.3 mm.

MPI-INF-3DHP. This is a more challenging dataset than Human3.6M dataset. In addition to MPJPE, we also compare on Percentage of Correct Keypoints (PCK) with a threshold of 150 mm and Area Under the Curve (AUC) for a range of PCK thresholds as alternate metrics for evaluation. The results of MPI-INF-3DHP without and with rigid alignment are presented in Table 5.2. Our method achieves the state-of-the-art performance on all metrics.

3DPW. The results are reported in Table 5.3. Our method has the second lowest MPJPE and PA-MPJPE, and is competitive against the regression method SPIN [**2**]. Our method also outperforms regression methods that use multiples frames [**165**, **166**].

Table 5.1. Evaluation on the Human3.6M dataset comparing computational times (s) and accuracy (mm) with Protocols 1 and 2. Overall, our method significantly outperforms all optimization methods with orders of magnitude speed up, and is competitive against the best performing regression method SPIN [**2**]. Preprocessing time for regression methods is the generation of human bounding boxes with YOLOv4-CSP [**3**], and for optimization methods is the inference time of the front-end neural network. All the optimization is run on CPU. VNect, MTC and ours are in C++, and SMPLify and UP-P91 are in Python.

	Method	Time (s)				Protocol 1		Protocol 2
		Preprocessing	Optimization	Regression	Total	MPJPE ↓	PA-MPJPE ↓	PA-MPJPE ↓
Pose only	Rogez et al. [**160**]	–	n/a	–	–	–	–	87.3
	Rogez et al. [**161**]	–	n/a	–	–	87.7	71.6	–
	Pavlakos et al. [**157**]	–	n/a	–	–	71.9	51.2	51.9
	Martinez et al. [**159**]	–	n/a	–	–	–	–	**47.7**
	Pavllo et al. [**158**]	–	n/a	–	–	**51.8**	**40**	–
	*VNect [**138**]	0.026	0.008	n/a	0.034	80.5	–	–
Pose and shape	HMR [**141**]	0.017	n/a	0.032	0.049	88.0	58.1	56.8
	Kolotouros et al. [**145**]	0.017	n/a	0.023	0.040	74.7	51.9	50.1
	SPIN [**2**]	0.017	n/a	**0.012**	**0.029**	**65.6**	**44.6**	**41.1**
	*SMPLify [**12**]	0.029	45	n/a	45	–	–	82.3
	*UP-P91 [**139**]	0.029	40	n/a	40	–	–	80.7
	*MTC [**11**]	0.029	20	n/a	20	64.5	–	–
	*Ours	0.029	**0.004**	n/a	**0.033**	**61.5**	48.2	**46.3**

(*) optimization method (n/a) not applicable (–) unreported statistic

Table 5.2. Evaluation on the MPI-INF-3DHP dataset. Our method outperforms optimization (denoted by *) and regression methods over multiple accuracy metrics before and after rigid alignment.

Method	PCK ↑	AUC ↑	MPJPE ↓
Absolute (w/o rigid alignment)			
Mehta et al. [**12**]	75.7	39.3	117.6
HMR [**141**]	72.9	36.5	124.2
SPIN [**2**]	76.4	37.1	105.2
*XNect [**137**]	77.8	38.9	115.0
*VNect [**138**]	76.6	40.4	124.7
*Ours	**83.0**	**41.9**	**91.5**
Rigid aligned			
HMR [**141**]	86.3	47.8	89.8
SPIN [**2**]	92.5	55.6	67.5
*VNect [**138**]	83.9	47.3	98.0
*Ours	**94.6**	**59.0**	**62.1**

Table 5.3. Evaluation on the 3DPW dataset. Our method is competitive against the best regression method SPIN. * denotes optimization method and ‡ indicates that the method uses multiple frames.

Method	MPJPE ↓	PA-MPJPE ↓
HMR [**141**]	130	81.3
Kolotouros et al. [**145**]	–	70.2
SPIN [**2**]	**96.9**	**59.2**
‡Arnab et al. [**166**]	–	72.2
‡Kanazawa et al. [**165**]	116.5	72.6
*XNect [**137**]	134.2	80.3
*Ours	98.6	68.0

5.6.4. Qualitative Results

We present typical failure cases due to inaccurate detection of our preprocessing pipeline in Fig. 5.3 and qualitative comparisons with SPIN [**2**] and SMPLify [**12**] on difficult examples from the Human3.6M [**13**], MPI-INF-3DHP [**14**] and 3DPW [**10**] datasets

Figure 5.3. Typical failure cases of our method due to (left) body part occlusion, (middle) incorrect body orientation detection, (right) depth ambiguity of monocular camera.

in Figs. 5.4 to 5.9. For a fair comparison, we add extra 3D keypoint measurements to SMPLify to improve its performance. In Figs. 5.4 to 5.9, it can be seen that our method has better pixel alignment than SPIN [**2**] and generates results of higher quality than SMPLify [**12**].

5.7. Ablation Studies

In the ablation stuidies, we analyze the impacts of the number of joints K, the number of measurements N, and the number of shape parameters P on the computation of the Gauss-Newton direction. The SMPL model [**144**] with $K = 23$ joints and SMPL+H model [**167**] with $K = 51$ joints are used for evaluation.

Figure 5.4. Qualitative comparisons of our method (second row in pink), SPIN **[2]** (third row in gray), and SMPLify **[12]** (fourth row in purple) on the Human3.6M **[13]** dataset.

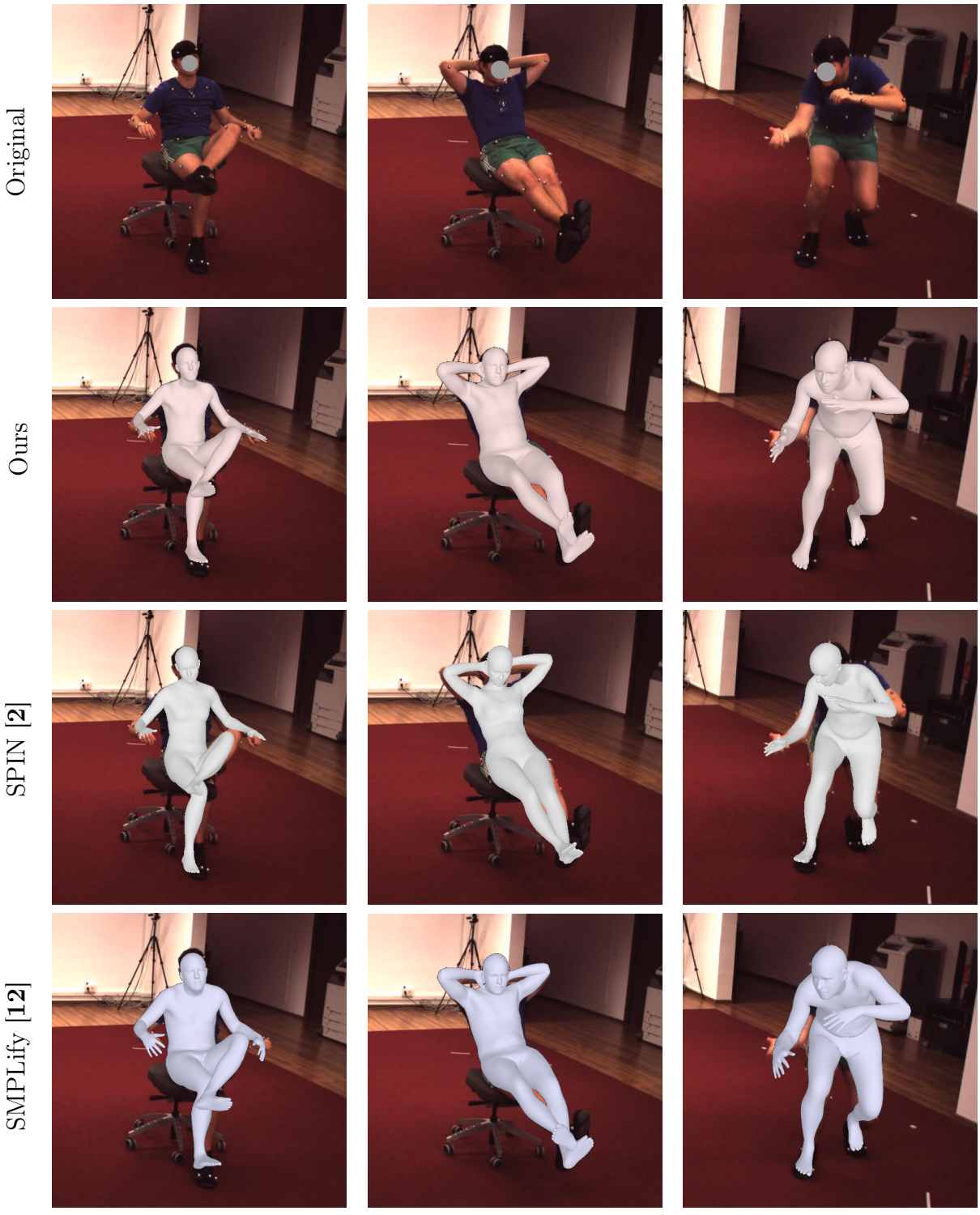


Figure 5.5. Qualitative comparisons of our method (second row in pink), SPIN **[2]** (third row in gray), and SMPLify **[12]** (fourth row in purple) on the Human3.6M **[13]** dataset.

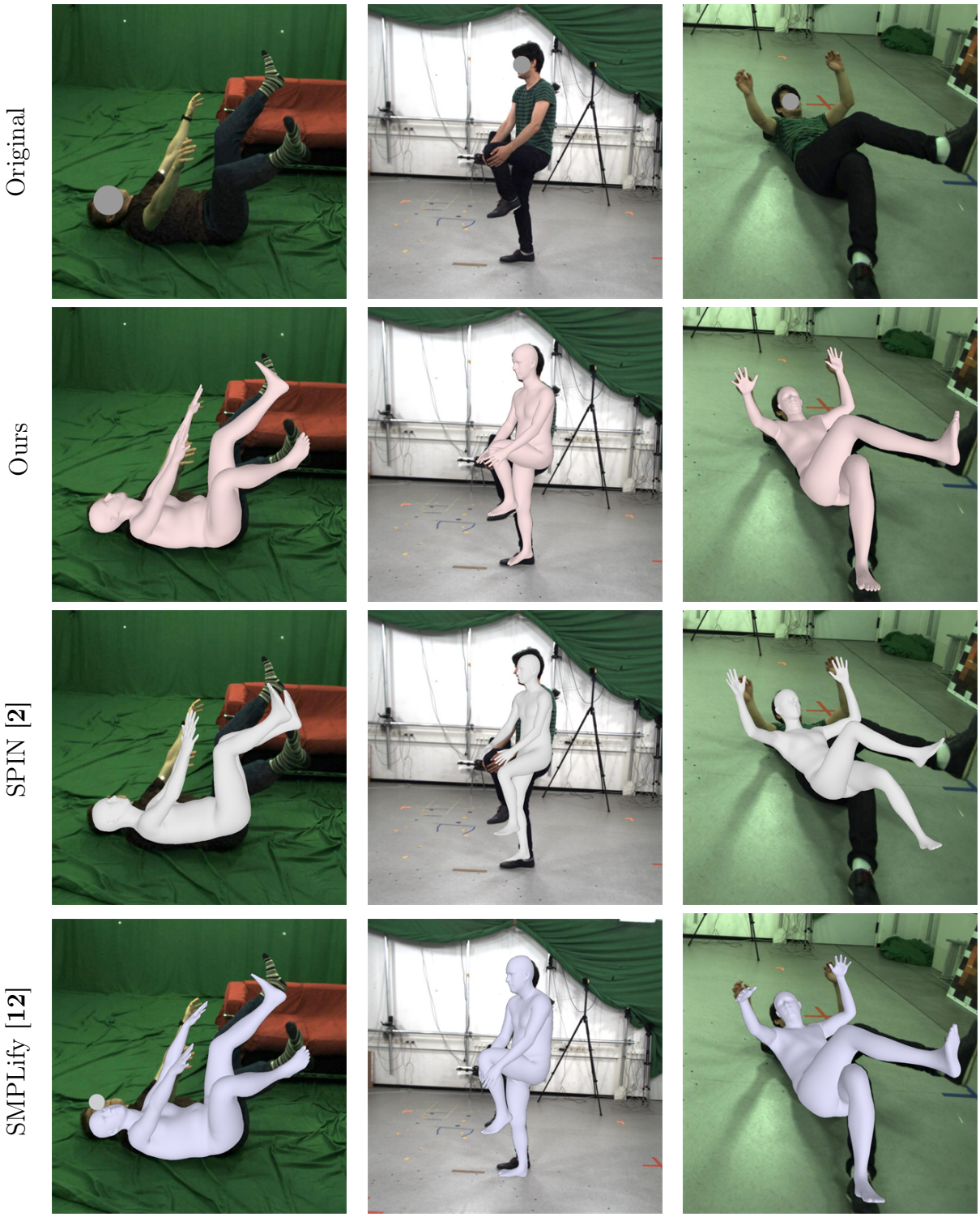


Figure 5.6. Qualitative comparisons of our method (second row in pink), SPIN [**2**] (third row in gray), and SMPLify [**12**] (fourth row in purple) on the MPI-INF-3DHP [**14**] dataset.

Figure 5.7. Qualitative comparisons of our method (second row in pink), SPIN [**2**] (third row in gray), and SMPLify [**12**] (fourth row in purple) on the MPI-INF-3DHP [**14**] dataset.

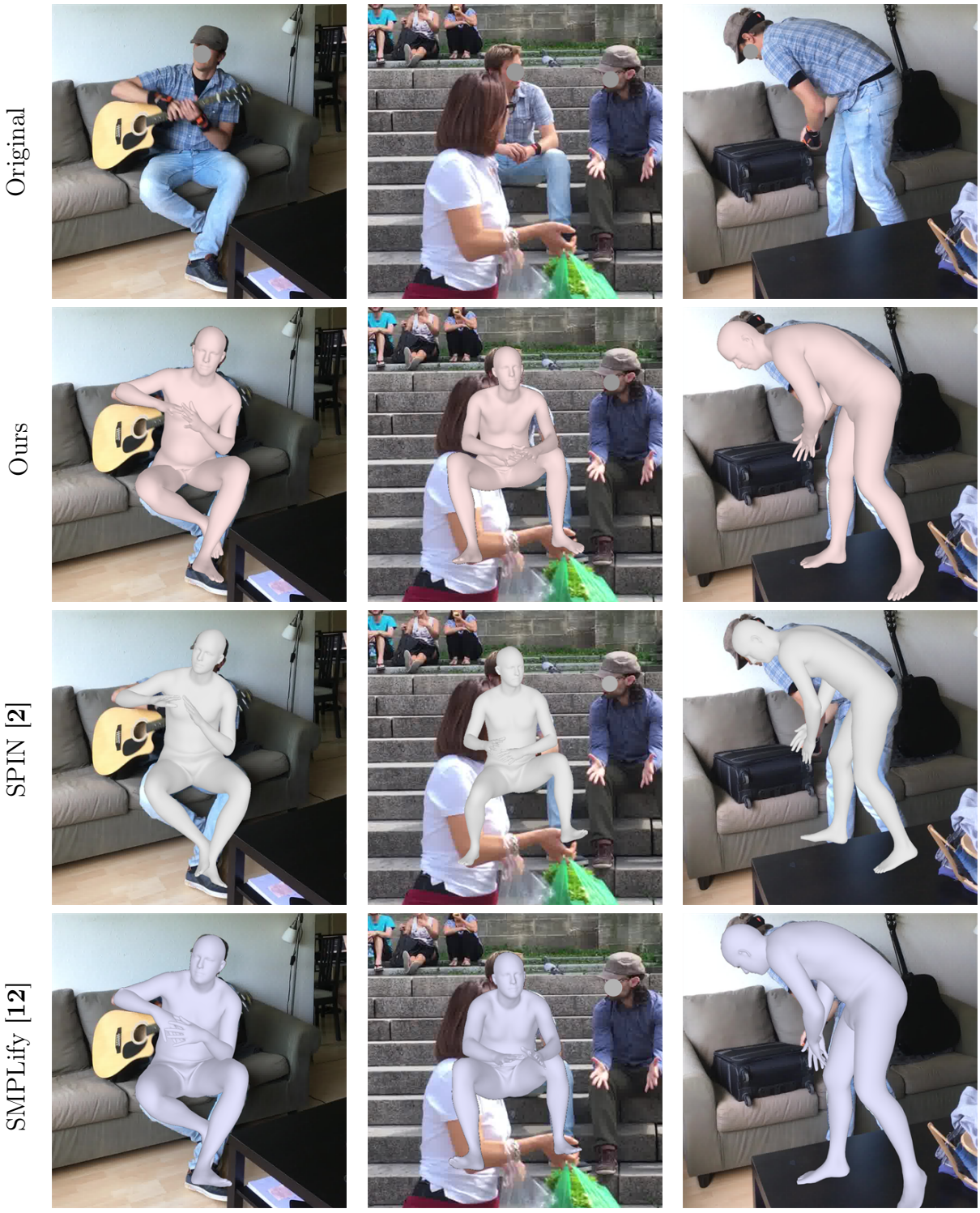


Figure 5.8. Qualitative comparisons of our method (second row in pink), SPIN [**2**] (third row in gray), and SMPLify [**12**] (fourth row in purple) on the 3DPW [**10**] dataset.

Figure 5.9. Qualitative comparisons of our method (second row in pink), SPIN [**2**] (third row in gray), and SMPLify [**12**] (fourth row in purple) on the 3DPW [**10**] dataset.

5.7.1. Experiments

In this section, the CPU time to compute the Gauss-Newton direction w/ and w/o our method is recorded for the SMPL and SMPL+H models in the following experiments.

Experiment 1. The number of shape parameters P is 0 and the number of measurements N increases from 120 to 600 for both of the SMPL and SMPL+H models.

Experiment 2. The number of shape parameters P is 10 and the number of measurements N increases from 120 to 600 for both of the SMPL and SMPL+H models.

Experiment 3. The number of shape parameters P increases from 0 to 10, and each joint of the SMPL and SMPL+H models is assigned with a 2D keypoint, a 3D keypoint, and a part orientation field as measurements.

The SMPL and SMPL+H models have different numbers of joints, and Experiments 1 to 3 have varying numbers of measurements and shape parameters. Thus, these experiments are sufficient to evaluate the impacts of the number of joints K, measurements N and shape parameters P on the computation of the Gauss-Newton direction.

5.7.2. Number of the Joints

The CPU time ratio of the SMPL+H and SMPL models to compute the Gauss-Newton direction is used as the metric to evaluate the impact of the number of joints K. Note that the SMPL and SMPL+H models have $K = 23$ and $K = 51$ joints, respectively. The CPU time ratio reflects the additional time induced as a result of the more joints on the SMPL+H model. The CPU time ratios of the three experiments are reported in Fig. 5.10 and discussed as follows:

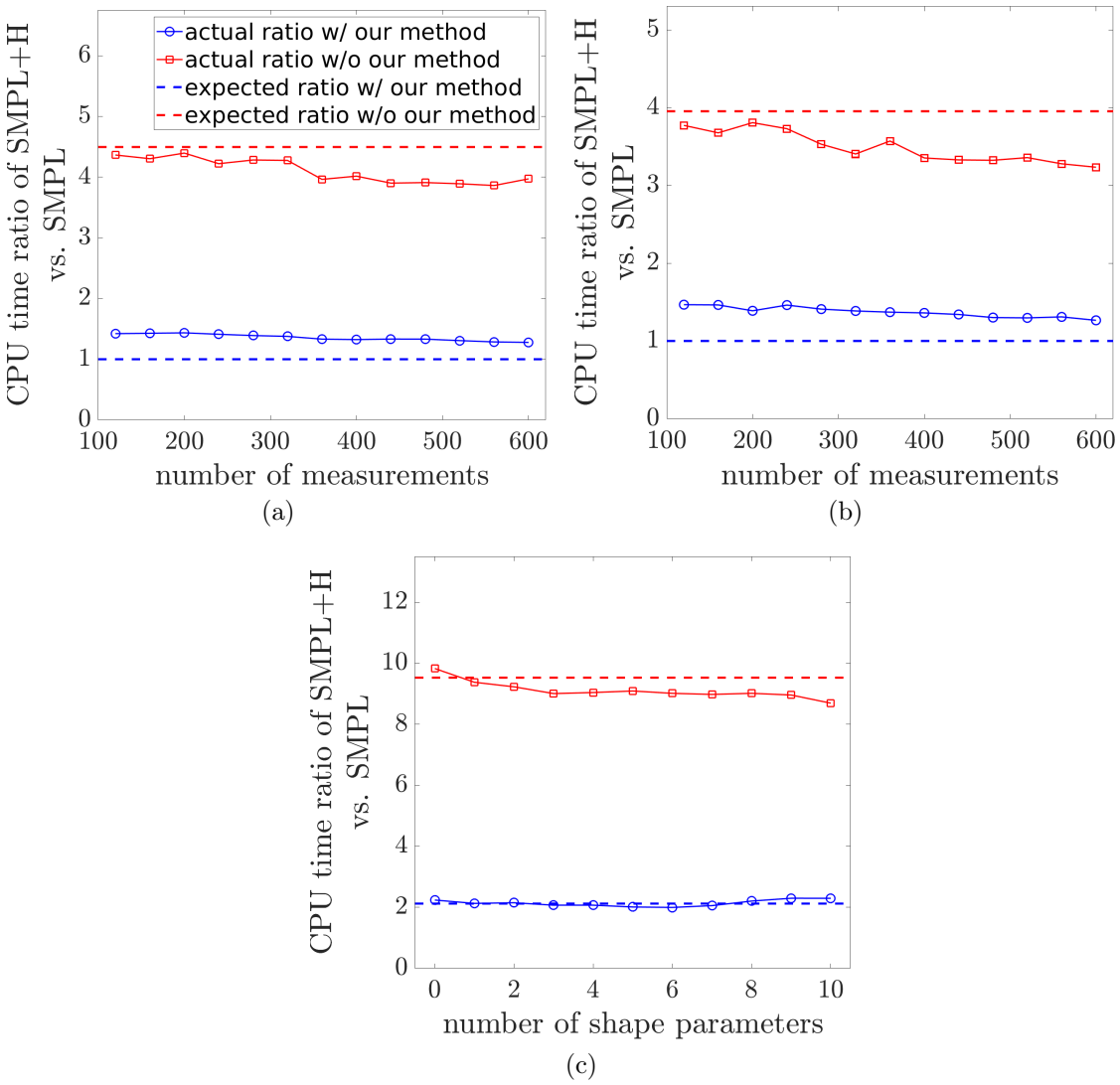


Figure 5.10. The CPU time ratio of the SMPL+H and SMPL models to compute the Gauss-Newton direction with (a) different numbers of measurements and no shape parameters, (b) different numbers of measurements and 10 shape parameters, and (c) different numbers of shape parameters. The SMPL and SMPL+H models have $K = 23$ and $K = 51$ joints, respectively. In Figs. 5.10 (a) to 5.10(c), the solid lines denote the actual CPU time ratio of the SMPL+H and SMPL models that is obtained from the experiments, whereas the dashed lines denote the expected CPU time ratio that is approximated from the complexity analysis in Tables 5.6 to 5.8. It can be seen the impact of the number of joints is around two orders of magnitude less on our method.

(1) In Experiment 1, there are no shape parameters and the computation of the Gauss-Newton direction is dominated by the number of measurements N. From Tables 5.6 to 5.8, it is known that our method has $O(N)$ complexity, which is not related with the number of joints K, and thus, the expected CPU time ratio with our method should be

$$\frac{1}{1} = 1.$$

In contrast, the CPU time without our method is approximately $O\left((3K+6)^2\right)$, which suggests an expected CPU time ratio of

$$\left(\frac{3 \times 51 + 6}{3 \times 23 + 6}\right)^2 = 4.49.$$

The numbers of 1 and 4.49 in the two equations above are consistent with the results in Fig. 5.10(a).

(2) In Experiment 2, there are 10 shape parameters. However, the analysis is still similar to that of Experiment 1. From Tables 5.6 to 5.8, the expected CPU time ratio of the SMPL+H and SMPL models w/ and w/o our method should be around

$$\frac{1}{1} = 1$$

and

$$\left(\frac{3 \times 51 + 6 + 10}{3 \times 23 + 6 + 10}\right)^2 = 3.95,$$

respectively, which is consistent with the results in Fig. 5.10(b).

(3) In Experiment 3, the number of measurements N is proportional to the number of joints of the SMPL and SMPL+H models. Then, as a result of Tables 5.6 to 5.8,

the CPU time w/ and w/o our method to compute the Gauss-Newton direction should be around $O(K)$ and $O\left((3K+6)^3\right)$, respectively, and the corresponding expected CPU time can be also approximated by

$$\frac{51}{23} = 2.22$$

and

$$\left(\frac{3\times 51+6}{3\times 23+6}\right)^3 = 9.53,$$

which is consistent with the results in Fig. 5.10(c).

(4) From Fig. 5.10 and the discussions above, it can be further concluded that the number of joints has around $O(K^2)$ times less impact on our method, which suggests that our sparse constrained formulation is more suitable for human models with more joints.

5.7.3. Number of the Measurements

The CPU time w/ and w/o our method to compute the Gauss-Newton direction and the corresponding speedup in Experiments 1 and 2 are reported in Figs. 5.11 and 5.12. It can be seen from Figs. 5.11 and 5.12 that our method has 4.73 $\sim$ 13.91x speedup on the SMPL model and a 12.17 $\sim$ 43.24x speedup on the SMPL+H model. Furthermore, no matter whether there are shape parameters or not, the speedup of our method is greater as the number of measurements increases, which means that our sparse constrained formulation is more efficient to solve optimization problems with more more measurements.

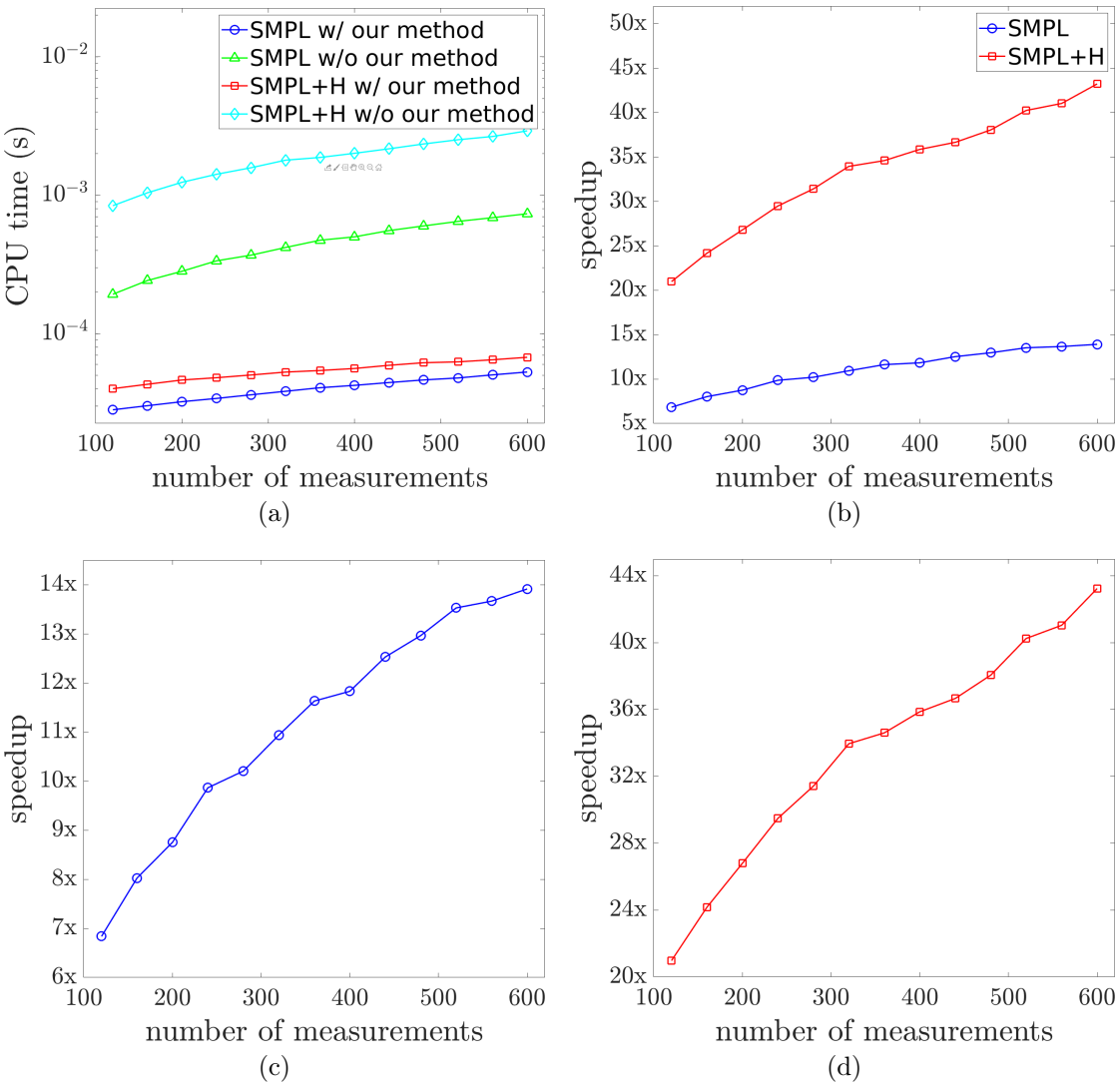


Figure 5.11. The computation of the Gauss-Newton direction with different numbers of measurements and no shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

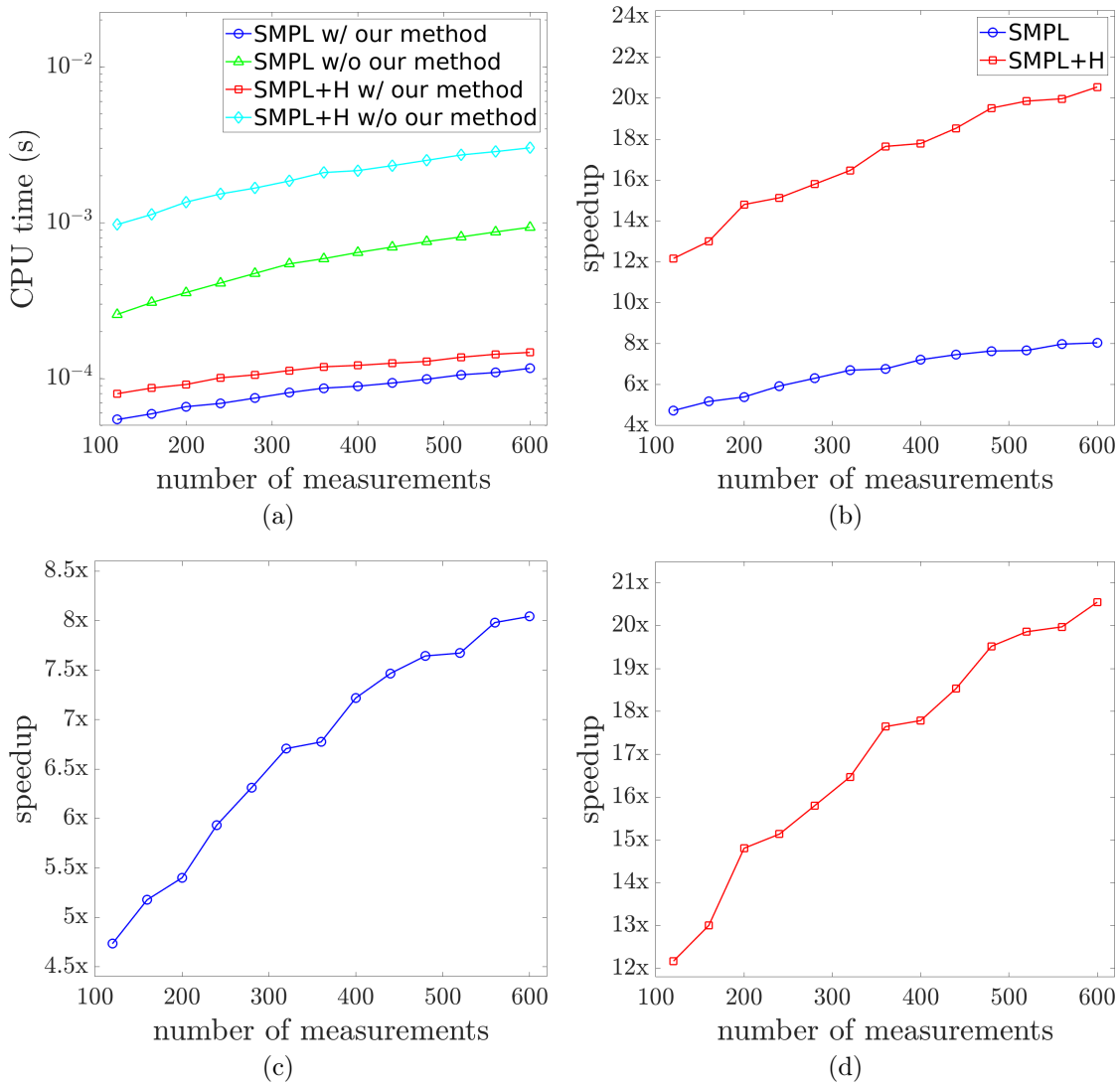


Figure 5.12. The computation of the Gauss-Newton direction with different numbers of measurements and 10 shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

5.7.4. Number of the Shape Parameters

The CPU time w/ and w/o our method to compute the Gauss-Newton direction and the corresponding speedup in Experiment 3 are reported in Fig. 5.13. It can be seen from Fig. 5.13 that our method has a 4.92 $\sim$ 7.78x speedup on the SMPL model and a 18.63 $\sim$ 34.18x speedup on the SMPL+H model, which is consistent with the analysis that our sparse constrained formulation has better scalability on human models with more joints. On the SMPL+H model, the CPU time taken to compute the Gauss-Newton direction without our method is as many as 2.5 ms, which is difficult to be used in real time considering that most optimization methods need around 20 $\sim$ 30 iterations to converge. As a comparison, our method is significantly faster on both of the SMPL and SMPL+H models, for which the CPU time is 0.027 $\sim$ 0.13 ms. In particular, note that if there are no shape parameters, our method has a further acceleration of the computation—this has is important for real-time video tracking of 3D human pose and shape, where the shape parameters that are estimated from the first few frames can be reused.

5.8. Conclusion

We considered the problem of 3D human pose and shape estimation by presenting a sparse constrained formulation that performs on par with regression methods. We demonstrated how to exploit the sparsity in our formulation and build an optimizer that can compute the Gauss-Newton direction in only linear complexity (with respect to the number of joints and measurements in the human model). This was a key contributing factor in bringing down the computation times of existing optimization methods by orders of

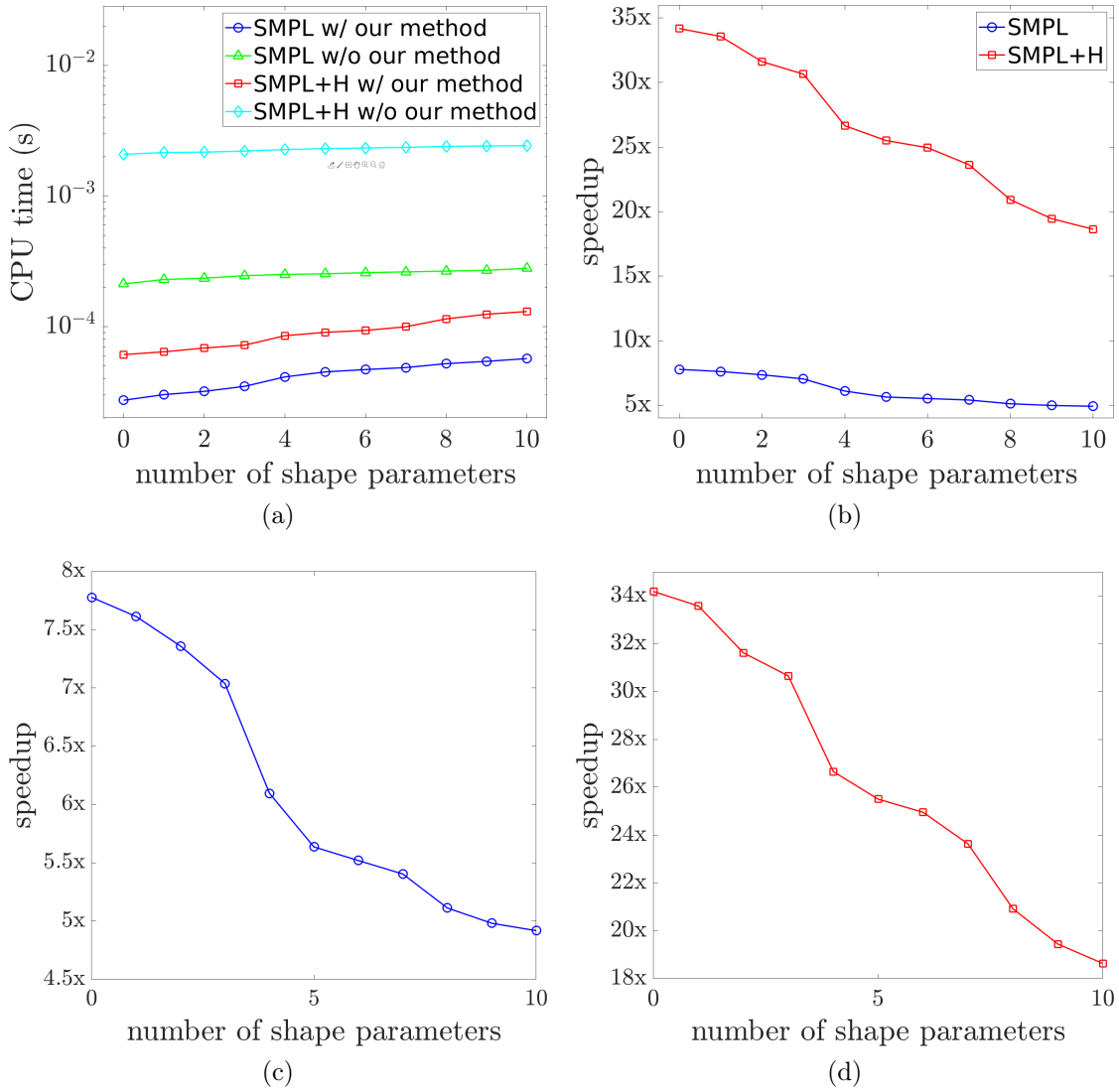


Figure 5.13. The computation of the Gauss-Newton direction with different number of shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

magnitude to 4ms. In benchmarks across multiple datasets on several metrics our framework, that uses a preprocessing neural network plus our optimizer, was highly competitive against the best performing regression method in terms of speed and accuracy.

We note that our fast framework can also benefit regression methods by quickly refining their outputs or by reducing training times for methods that train with some optimization in the loop.

The qualitative results illustrate that our framework was mainly limited by the reliability of the preprocessor. While our primary focus in this work was on the optimization side, some investment in engineering the preprocessor could yield further improvements in performance. Although we employed the SMPL model in our current implementation, our optimizer has the flexibility to support other types of 3D human models if the appropriate loss terms are specified for the objective. In particular, sparse 3D human models such as STAR [**147**] would be well suited for our method. With an additional preprocessor, and model and loss terms to support human hands and facial expressions, our framework can also be extended to address the total 3D human capture problem.

5.9. Proofs

5.9.1. Proof of Proposition 5.4.1

In Eqs. (5.16) and (5.19), $\mathbf{T}_i \in SE(3)$ is the rigid body transformation of body part i, and $\boldsymbol{\Omega}_i$ is the state of joint i, and $\boldsymbol{\Omega} \triangleq (\boldsymbol{\Omega}_1, \cdots, \boldsymbol{\Omega}_K) \in SO(3)^K$ are the joint states, and $\boldsymbol{\beta}$ and $\boldsymbol{\beta}_i \in \mathbb{R}^P$ are the shape parameters, and $\mathtt{F}_i(\cdot) : SE(3) \times \mathbb{R}^P \times SO(3) \rightarrow SE(3)$ is a function that maps $\mathbf{T}_{\text{par}(i)}$, $\boldsymbol{\beta}_{\text{par}(i)}$ and $\boldsymbol{\Omega}_i$ to $\mathbf{T}_i$.

Note that Eqs. (5.17) and (5.18) are equivalent to Eqs. (5.19) and (5.20). If we let $\boldsymbol{\beta}_0 = \boldsymbol{\beta}$, Eq. (5.18b) suggests that $\boldsymbol{\beta}_i = \boldsymbol{\beta}$ for all $i = 1, \cdots, K$, from which Eqs. (5.19) and (5.20) are reduced to

$$\min_{\{\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta}_i\}_{i=0}^{K}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta})\|^2 \tag{5.27}$$

subject to

$$\begin{aligned} \mathbf{T}_i =& \mathtt{F}_i(\mathbf{T}_{\mathrm{par}(i)}, \boldsymbol{\beta}, \boldsymbol{\Omega}_i) \\ =& \mathbf{T}_{\mathrm{par}(i)} \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}. \end{aligned} \tag{5.28}$$

Next, as mentioned in Section 5.3.4, if we perform a top-down traversal of the kinematic tree of the SMPL model and recursively exploit Eq. (5.28) for each body part $i = 1, \cdots, K$, then, all of $\mathbf{T}_i \in SE(3)$ can be represented as a function of the root pose $\mathbf{T}_0 \in SE(3)$, and the joint states $\boldsymbol{\Omega} \in SO(3)^K$, and the shape parameter $\boldsymbol{\beta} \in \mathbb{R}^P$, i.e.,

$$\mathbf{T}_i \triangleq \mathbf{T}_i\left(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}\right) \tag{5.29}$$

If we use Eq. (5.29) to cancel out non-root rigid body transformations $\mathbf{T}_i$ $(1 \leq i \leq K)$, each $\mathtt{r}_i(\cdot)$ in Eq. (5.27) is rewritten as a function of $\mathbf{T}_0 \in SE(3)$, and $\boldsymbol{\Omega} \in SO(3)^K$, and $\boldsymbol{\beta} \in \mathbb{R}^P$, from which we obtain an optimization problem of a dense unconstrained formulation

$$\min_{\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta})\|^2$$

that is the same as Eq. (5.16). Therefore, it can be concluded that Eqs. (5.19) and (5.20) are equivalent to Eq. (5.16). The proof is completed.

5.9.2. Proof of Proposition 5.4.2

The proof of Proposition 5.4.2 is organized as follows: we present an overview of the steps to compute the Gauss-Newton direction in Section 5.9.2.1, and show that the steps for the two formulations result in the same Gauss-Newton direction in Section 5.9.2.2, and derive a dynamic programming algorithm to solve the quadratic program of the sparse constrained formulation in Section 5.9.2.3, and analyze the complexity of the aforementioned steps to compute the Gauss-Newton direction in Section 5.9.2.4.

5.9.2.1. Steps to Compute the Gauss-Newton Direction. We introduce $\mathbf{x} \triangleq (\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}) \in SE(3) \times SO(3)^K \times \mathbb{R}^P$ and $\mathbf{x}_i \triangleq (\mathbf{T}_i, \boldsymbol{\beta}_i) \in SE(3) \times \mathbb{R}^P$. Then Eq. (5.16) and Eqs. (5.19) and (5.20) can be rewritten as

$$\min_{\mathbf{x}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{x})\|^2, \tag{5.30}$$

and

$$\min_{\{\mathbf{x}_i, \boldsymbol{\Omega}_i\}_{i=0}^{K}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{x}_i, \boldsymbol{\Omega}_i)\|^2 \tag{5.31}$$

subject to

$$\mathbf{x}_i = \begin{bmatrix} \mathtt{F}_i(\mathbf{x}_{\mathrm{par}(i)}, \boldsymbol{\Omega}_i) \\ \boldsymbol{\beta}_{\mathrm{par}(i)} \end{bmatrix}, \tag{5.32}$$

respectively. For analytical clarity, we assume with no loss of generality that the residues $\mathtt{r}_i(\mathbf{x})$ and $\mathtt{r}_i(\mathbf{x}_i, \boldsymbol{\Omega}_i)$ are $N_i \times 1$ vectors for $i = 0, \cdots, K$.

Following the procedure originally given in Section 5.4, an overview of steps to compute the Gauss-Newton direction for the dense unconstrained and sparse constrained formulations is given in Tables 5.4 and 5.5, which will be frequently used in the rest of this proof.

5.9.2.2. The Equivalence of the Gauss-Newton Direction. In Tables 5.4 and 5.5, since Steps 2 and 3 are the reformulation of Step 1, we only need to show that the linearizations of dense unconstrained and sparse constrained formulations in Step 1, i.e.,

Table 5.4. Steps to Compute the Gauss-Newton Direction for the Dense Unconstrained Formulation

Step 1	The linearization of Eq. (5.30) results in (5.33) $$\min_{\Delta\mathbf{x}} \Delta\mathtt{E} = \sum_{i=0}^{K} \frac{1}{2}\|\mathbf{J}_i\Delta\mathbf{x} + \mathtt{r}_i\|^2,$$ where $\Delta\mathbf{x} \triangleq (\Delta\mathbf{T}_0, \Delta\boldsymbol{\Omega}, \Delta\boldsymbol{\beta}) \in \mathbb{R}^{6+3K+P}$, $\Delta\mathbf{T}_0 \in \mathbb{R}^6$, $\Delta\boldsymbol{\Omega} \in \mathbb{R}^{3K}$ and $\Delta\boldsymbol{\beta} \in \mathbb{R}^P$ are the Gauss-Newton directions of $\mathbf{x}$, $\mathbf{T}_0$, $\boldsymbol{\Omega}$ and $\boldsymbol{\beta}$, respectively, and (5.34) $$\mathbf{J}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{r}_i}{\partial \mathbf{T}_0} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\Omega}} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\beta}} \end{bmatrix} \in \mathbb{R}^{N_i \times (6+3K+P)}$$ is the Jacobian of $\mathtt{r}_i(\cdot)$, and $\mathtt{r}_i \in \mathbb{R}^{N_i}$ is the residue.
Step 2	Reformulate Eq. (5.33) as (5.35) $$\min_{\Delta\mathbf{x}} \Delta\mathtt{E} = \frac{1}{2}\Delta\mathbf{x}^\top\mathbf{H}\Delta\mathbf{x} + \mathbf{g}^\top\Delta\mathbf{x}$$ where $\mathbf{H} \triangleq \sum_{i=0}^{K} \mathbf{J}_i^\top\mathbf{J}_i \in \mathbb{R}^{(6+3K+P)\times(6+3K+P)}$ is the Hessian, and $\mathbf{g} \triangleq \sum_{i=0}^{K} \mathbf{J}_i^\top\mathtt{r}_i \in \mathbb{R}^{(6+3K+P)}$ is the gradient.
Step 3	Compute the Gauss-Newton direction from Eq. (5.35), which has a closed-form solution (5.36) $$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}.$$

Table 5.5. Steps to Compute the Gauss-Newton Direction for the Sparse Constrained Formulation

Step 1

The linearization of Eq. (5.31) results in

$$\min_{\{\Delta\mathbf{x}_i,\,\Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}} \Delta\mathtt{E} = \sum_{i=0}^{K} \frac{1}{2}\|\mathbf{J}_{i,1}\Delta\mathbf{x}_i + \mathbf{J}_{i,2}\Delta\boldsymbol{\Omega}_i + \mathtt{r}_i\|^2 \tag{5.37}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i, \tag{5.38}$$

where $\Delta\mathbf{x}_i \triangleq (\Delta\mathbf{T}_i,\, \Delta\boldsymbol{\beta}_i) \in \mathbb{R}^{6+P}$, $\Delta\mathbf{T}_i \in \mathbb{R}^6$, $\Delta\boldsymbol{\Omega}_i \in \mathbb{R}^3$ and $\Delta\boldsymbol{\beta}_i \in \mathbb{R}^P$ are the Gauss-Newton directions of $\mathbf{x}_i$, $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$ and $\boldsymbol{\beta}_i$, respectively, and

$$\mathbf{J}_{i,1} \triangleq \begin{bmatrix} \dfrac{\partial \mathtt{r}_i}{\partial \mathbf{T}_i} & \dfrac{\partial \mathtt{r}_i}{\partial \boldsymbol{\beta}_i} \end{bmatrix} \in \mathbb{R}^{N_i\times(6+P)} \tag{5.39}$$

and

$$\mathbf{J}_{i,2} \triangleq \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\Omega}_i} \in \mathbb{R}^{N_i\times 3} \tag{5.40}$$

are the Jacobians of $\mathtt{r}_i(\cdot)$, and

$$\mathbf{A}_i \triangleq \begin{bmatrix} \dfrac{\partial \mathtt{F}_i}{\partial \mathbf{T}_{\mathrm{par}(i)}} & \dfrac{\partial \mathtt{F}_i}{\partial \boldsymbol{\beta}_{\mathrm{par}(i)}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{(6+P)\times(6+P)} \tag{5.41}$$

and

$$\mathbf{B}_i \triangleq \begin{bmatrix} \dfrac{\partial \mathtt{F}_i}{\partial \boldsymbol{\Omega}_i} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(6+P)\times 3} \tag{5.42}$$

are the partial derivatives of Eq. (5.38), and $\mathtt{r}_i \in \mathbb{R}^{N_i}$ is the residue.

Step 2

Reformulate Eq. (5.37) as

$$\begin{aligned}\min_{\{\Delta\mathbf{x}_i,\,\Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}} \Delta\mathtt{E} = \sum_{i=0}^{K} \Big[&\frac{1}{2}\Delta\mathbf{x}_i^\top\mathbf{H}_{i,11}\Delta\mathbf{x}_i + \Delta\boldsymbol{\Omega}_i^\top\mathbf{H}_{i,21}\Delta\mathbf{x}_i + \\ &\frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top\mathbf{H}_{i,22}\Delta\boldsymbol{\Omega}_i + \mathbf{g}_{i,1}^\top\Delta\mathbf{x}_i + \mathbf{g}_{i,2}^\top\Delta\boldsymbol{\Omega}_i\Big]\end{aligned} \tag{5.43}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i,$$

where $\mathbf{H}_{i,11} \triangleq \mathbf{J}_{i,1}^\top\mathbf{J}_{i,1} \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{H}_{i,21} \triangleq \mathbf{J}_{i,2}^\top\mathbf{J}_{i,1} \in \mathbb{R}^{3\times(6+P)}$, and $\mathbf{H}_{i,22} \triangleq \mathbf{J}_{i,2}^\top\mathbf{J}_{i,2} \in \mathbb{R}^{3\times 3}$ are the Hessians, and $\mathbf{g}_{i,1} \triangleq \mathbf{J}_{i,1}^\top\mathtt{r}_i \in \mathbb{R}^{6+P}$ and $\mathbf{g}_{i,2} \triangleq \mathbf{J}_{i,2}^\top\mathtt{r}_i \in \mathbb{R}^{6+P}$ are the gradients.

Step 3

Compute the Gauss-Newton direction from Eq. (5.43), which can be exactly solved by Algorithm 15.

Eqs. (5.33) and (5.37), are equivalent. From Eq. (5.29), the rigid body transformation $\mathbf{T}_i \in SE(3)$ of body part i can be written as a function of $\mathbf{T}_0$, $\boldsymbol{\Omega}$ and $\boldsymbol{\beta}$. Furthermore, it is by the definition of $\mathtt{r}_i(\cdot)$ that

$$\mathtt{r}_i(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}) = \mathtt{r}_i\big(\mathbf{T}_i(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}), \boldsymbol{\Omega}_i, \boldsymbol{\beta}\big).$$

From the equation above, $\mathbf{J}_i\Delta\mathbf{x}$ in Eq. (5.33) can be computed using $\mathbf{J}_{i,1}$ and $\mathbf{J}_{i,2}$ in Eq. (5.37):

$$\mathbf{J}_i\Delta\mathbf{x} = \mathbf{J}_{i,1}\begin{bmatrix} \frac{\partial \mathbf{T}_i}{\partial \mathbf{T}_0}\Delta\mathbf{T}_0 + \frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\Omega}}\Delta\boldsymbol{\Omega} + \frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\beta}}\Delta\boldsymbol{\beta} \\ \\ \boldsymbol{\beta} \end{bmatrix} + \mathbf{J}_{i,2}\Delta\boldsymbol{\Omega}_i. \tag{5.44}$$

Note that the partial derivatives $\frac{\partial \mathbf{T}_i}{\partial \mathbf{T}_0}$, $\frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\Omega}}$ and $\frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\beta}}$ in the right-hand side of Eq. (5.44) are obtained by the recursive implementation of Eq. (5.38). Therefore, it can be concluded that Eqs. (5.33) and (5.37) are equivalent to each other, which suggests that the dense unconstrained and sparse constrained formulations result in the same Gauss-Newton direction.

5.9.2.3. Algorithm to Solve Eq. (5.43). In Table 5.5, it is straightforward to follow **Steps 1–2** of the sparse constrained formulation to compute the Gauss-Newton direction. Next, we need to solve the quadratic program of Eq. (5.43) in **Step 3**, which is nontrivial. In this subsection, we derive a dynamic programming algorithm that exploits the sparsity and constraints of Eq. (5.38) such that the Gauss-Newton direction can be exactly computed.

For notational simplicity, we let $\mathrm{par}(i)$, $\mathrm{chd}(i)$ and $\mathrm{des}(i)$ be the parent, children and descendants of body part i in the kinematics tree, and assume $i > \mathrm{par}(i)$ for all $i = 1, \cdots, K$.

First, we define $\mathcal{E}_i(\cdot) : \mathbb{R}^{6+P} \to \mathbb{R}$ to be a function of $\Delta\mathbf{x}_{\mathrm{par}(i)} \in \mathbb{R}^{6+P}$ in the form of an optimization problem of $\{\Delta\mathbf{x}_j, \Delta\boldsymbol{\Omega}_j\}$ for $j \in \{i\} \cup \mathrm{des}(i)$

$$\begin{aligned}(5.45)\quad \mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) \triangleq \min_{\{\Delta\mathbf{x}_j,\, \Delta\boldsymbol{\Omega}_j\}_{j\in\{i\}\cup\mathrm{des}(i)}} \sum_{j\in\{i\}\cup\mathrm{des}(i)} \Big[&\frac{1}{2}\Delta\mathbf{x}_j^\top \mathbf{H}_{j,11}\Delta\mathbf{x}_j + \\ &\Delta\boldsymbol{\Omega}_j^\top \mathbf{H}_{j,21}\Delta\mathbf{x}_j + \frac{1}{2}\Delta\boldsymbol{\Omega}_j^\top \mathbf{H}_{j,22}\Delta\boldsymbol{\Omega}_j + \mathbf{g}_{j,1}^\top\Delta\mathbf{x}_j + \mathbf{g}_{j,2}^\top\Delta\boldsymbol{\Omega}_j\Big]\end{aligned}$$

subject to

$$(5.46)\qquad \Delta\mathbf{x}_j = \mathbf{A}_j\Delta\mathbf{x}_{\mathrm{par}(j)} + \mathbf{B}_j\Delta\boldsymbol{\Omega}_j,\ \forall j \in \{i\} \cup \mathrm{des}(i),$$

where $\Delta\mathbf{x}_{\mathrm{par}(i)} \in \mathbb{R}^{6+P}$ is given. Furthermore, if $\mathcal{E}_j(\cdot) : \mathbb{R}^{6+P} \to \mathbb{R}$ is defined for all $j \in \mathrm{chd}(i)$, then, it is from Eq. (5.45) that $\mathcal{E}_i(\cdot)$ can be reduced to an optimization problem of $\Delta\mathbf{x}_i$ and $\Delta\boldsymbol{\Omega}_i$

$$\begin{aligned}(5.47)\quad \mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) \triangleq \min_{\Delta\mathbf{x}_i,\, \Delta\boldsymbol{\Omega}_i} \Big[&\frac{1}{2}\Delta\mathbf{x}_i^\top \mathbf{H}_{i,11}\Delta\mathbf{x}_i + \Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,21}\Delta\mathbf{x}_i + \\ &\frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,22}\Delta\boldsymbol{\Omega}_i + \mathbf{g}_{i,1}^\top\Delta\mathbf{x}_i + \mathbf{g}_{i,2}^\top\Delta\boldsymbol{\Omega}_i + \sum_{j\in\mathrm{chd}(i)} \mathcal{E}_j(\Delta\mathbf{x}_i)\Big]\end{aligned}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i,$$

where $\Delta\mathbf{x}_{\mathrm{par}(i)} \in \mathbb{R}^{6+P}$ is given. Note that Eq. (5.47) is an intermediate procedure that is essential for our dynamic programming algorithm.

Next, suppose that there exists $\mathbf{M}_j \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_j \in \mathbb{R}^{6+P}$ and $\Delta\mathtt{E}_j \in \mathbb{R}$ for all $j \in \mathrm{chd}(i)$ such that $\mathcal{E}_j(\Delta\mathbf{x}_i)$ can be written as

$$\mathcal{E}_j(\Delta\mathbf{x}_i) = \frac{1}{2}\Delta\mathbf{x}_i^\top \mathbf{M}_j \Delta\mathbf{x}_i + \mathbf{m}_j^\top \Delta\mathbf{x}_i + \Delta\mathtt{E}_j. \tag{5.48}$$

Applying Eq. (5.48) to Eq. (5.47), we obtain

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \min_{\Delta\mathbf{x}_i,\,\Delta\boldsymbol{\Omega}_i} \frac{1}{2}\Delta\mathbf{x}_i \mathbf{N}_{i,11}\Delta\mathbf{x}_i + \Delta\boldsymbol{\Omega}_i^\top \mathbf{N}_{i,21}\Delta\mathbf{x}_i + \\ \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top \mathbf{N}_{i,22}\Delta\boldsymbol{\Omega}_i + \mathbf{n}_{i,1}^\top \Delta\mathbf{x}_i + \mathbf{n}_{i,2}^\top \Delta\boldsymbol{\Omega}_i + \Delta\bar{\mathtt{E}}_i\end{aligned} \tag{5.49}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i \Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i \Delta\boldsymbol{\Omega}_i,$$

where

$$\mathbf{N}_{i,11} = \mathbf{H}_{i,11} + \sum_{j\in\mathrm{chd}(i)} \mathbf{M}_i, \tag{5.50a}$$

$$\mathbf{N}_{i,21} = \mathbf{H}_{i,21}, \tag{5.50b}$$

$$\mathbf{N}_{i,22} = \mathbf{H}_{i,22}, \tag{5.50c}$$

$$\mathbf{n}_{i,1} = \mathbf{g}_{i,1} + \sum_{j\in\mathrm{chd}(i)} \mathbf{m}_j, \tag{5.50d}$$

$$\mathbf{n}_{i,2} = \mathbf{g}_{i,2}, \tag{5.50e}$$

$$\Delta\overline{\mathtt{E}}_i = \sum_{j \in \mathrm{chd}(i)} \Delta\mathtt{E}_j. \tag{5.50f}$$

Substitute Eq. (5.38) into Eq. (5.49) to cancel out $\Delta\mathbf{x}_i$ and simplify the resulting equation to an unconstrained optimization problem on $\Delta\boldsymbol{\Omega}_i \in \mathbb{R}^3$:

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \min_{\Delta\boldsymbol{\Omega}_i} \frac{1}{2}\Delta\mathbf{x}_{\mathrm{par}(i)}\mathbf{Q}_{i,11}\Delta\mathbf{x}_{\mathrm{par}(i)} + \Delta\boldsymbol{\Omega}_i^\top\mathbf{Q}_{i,21}\Delta\mathbf{x}_{\mathrm{par}(i)} + \\ \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top\mathbf{Q}_{i,22}\Delta\boldsymbol{\Omega}_i + \mathbf{q}_{i,1}^\top\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{q}_{i,2}^\top\Delta\boldsymbol{\Omega}_i + \Delta\overline{\mathtt{E}}_i,\end{aligned} \tag{5.51}$$

where

$$\mathbf{Q}_{i,11} = \mathbf{A}_i^\top\mathbf{N}_{i,11}\mathbf{A}_i, \tag{5.52a}$$

$$\mathbf{Q}_{i,21} = \mathbf{B}_i^\top\mathbf{N}_{i,11}\mathbf{A}_i + \mathbf{N}_{i,21}\mathbf{A}_i, \tag{5.52b}$$

$$\mathbf{Q}_{i,22} = \mathbf{B}_i^\top\mathbf{N}_{i,11}\mathbf{B}_i + \mathbf{N}_{i,21}\mathbf{B}_i + \mathbf{B}_i^\top\mathbf{N}_{i,21}^\top + \mathbf{N}_{i,22}, \tag{5.52c}$$

$$\mathbf{q}_{i,1} = \mathbf{A}_i^\top\mathbf{n}_{i,1}, \tag{5.52d}$$

$$\mathbf{q}_{i,2} = \mathbf{B}_i^\top\mathbf{n}_{i,1} + \mathbf{n}_{i,2}. \tag{5.52e}$$

It is obvious that Eq. (5.51) has a closed-form solution

$$\Delta\boldsymbol{\Omega}_i = \mathbf{K}_i \Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{k}_i, \tag{5.53}$$

where

$$\mathbf{K}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21} \quad \text{and} \quad \mathbf{k}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}. \tag{5.54}$$

If we use Eq. (5.53) to eliminate $\Delta\boldsymbol{\Omega}_i$ in Eq. (5.51), there exists $\mathbf{M}_i \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_i \in \mathbb{R}^{6+P}$ and $\Delta\mathtt{E}_i \in \mathbb{R}$ such that

$$\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \frac{1}{2}\Delta\mathbf{x}_{\mathrm{par}(i)}^{\top}\mathbf{M}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{m}_i^{\top}\Delta\mathbf{x}_{\mathrm{par}(i)} + \Delta\mathtt{E}_i, \tag{5.55}$$

where

$$\mathbf{M}_i = \mathbf{Q}_{i,11} - \mathbf{Q}_{i,21}^{\top}\mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21}, \tag{5.56a}$$

$$\mathbf{m}_i = \mathbf{q}_{i,1} - \mathbf{Q}_{i,21}^{\top}\mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}, \tag{5.56b}$$

$$\Delta\mathtt{E}_i = \Delta\overline{\mathtt{E}}_i - \frac{1}{2}\mathbf{q}_{i,2}^{\top}\mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}. \tag{5.56c}$$

Therefore, if there exists $\mathbf{M}_j \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_j \in \mathbb{R}^{6+P}$ and $\Delta\mathtt{E}_j \in \mathbb{R}$ for all $j \in \mathrm{chd}(i)$ such that Eq. (5.48) holds, we might further obtain $\mathbf{M}_i \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_i \in \mathbb{R}^{6+P}$ and $\Delta\mathtt{E}_i \in \mathbb{R}$ with which $\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)})$ can be written as Eq. (5.55).

In the kinematic tree, a body part i at the leaf node has no children, for which Eq. (5.50) is simplified to $\mathbf{N}_{i,11} = \mathbf{H}_{i,11}$, $\mathbf{N}_{i,21} = \mathbf{H}_{i,21}$, $\mathbf{N}_{i,22} = \mathbf{H}_{i,22}$, $\mathbf{n}_{i,1} = \mathbf{g}_{i,1}$, $\mathbf{n}_{i,2} = \mathbf{g}_{i,2}$ and $\Delta\overline{\mathtt{E}}_i = 0$, then, it is possible to recursively compute $\mathbf{M}_i \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_i \in \mathbb{R}^{6+P}$ and $\Delta\mathtt{E}_i \in \mathbb{R}$ for each $i = 1, \cdots, K$ following Eqs. (5.50), (5.52) and (5.56) through the bottom-up traversal of kinematic tree.

It is by definition that $\boldsymbol{\Omega}_0$ is a dummy variable and $\Delta\boldsymbol{\Omega}_0 = \mathbf{0}$. Thus, if $\mathcal{E}_i(\Delta\mathbf{x}_0)$ in Eq. (5.55) is known for each $i \in \mathrm{chd}(0)$, Eq. (5.43) is equivalent to an unconstrained optimization problem on $\Delta\mathbf{x}_0 \in \mathbb{R}^{6+P}$:

$$\min_{\Delta\mathbf{x}_0} \frac{1}{2}\Delta\mathbf{x}_0^\top \mathbf{H}_{0,11}\Delta\mathbf{x}_0 + \mathbf{g}_{0,1}^\top\Delta\mathbf{x}_0 + \sum_{j\in\mathrm{chd}(0)} \mathcal{E}_i(\Delta\mathbf{x}_0).$$

From Eq. (5.55), the equation above is equivalent to

$$\min_{\Delta\mathbf{x}_0} \frac{1}{2}\Delta\mathbf{x}_0^\top \mathbf{M}_0\Delta\mathbf{x}_0 + \mathbf{m}_0^\top\mathbf{x}_0 + \Delta\overline{\mathtt{E}}_0 \tag{5.57}$$

where

$$\mathbf{M}_0 = \mathbf{H}_{0,11} + \sum_{j\in\mathrm{chd}(0)} \mathbf{M}_j, \tag{5.58a}$$

$$\mathbf{m}_0 = \mathbf{g}_{0,1} + \sum_{j\in\mathrm{chd}(0)} \mathbf{m}_j, \tag{5.58b}$$

$$\Delta\overline{\mathtt{E}}_0 = \sum_{i\in\mathrm{chd}(0)} \Delta\mathtt{E}_i. \tag{5.58c}$$

It is straightforward to show that

$$\Delta \mathbf{x}_0 = -\mathbf{M}_0^{-1}\mathbf{m}_0 \tag{5.59}$$

solves Eq. (5.57) with

$$\Delta \mathtt{E}_0 = \Delta \bar{\mathtt{E}}_0 - \frac{1}{2}\mathbf{m}_0^\top \mathbf{M}_0^{-1}\mathbf{m}_0 \tag{5.60}$$

to be the expected cost reduction as well as the minimum objective value of Eq. (5.43).

At last, we recursively compute $\{\Delta \mathbf{x}_i, \Delta \boldsymbol{\Omega}_i\}_{i=1}^K$ using Eqs. (5.38), (5.53) and (5.54) through a top-down traversal of the kinematics tree, from which the Gauss-Newton direction is exactly retrieved.

From our analysis, the resulting algorithm to solve Eq. (5.43) and compute the Gauss-Newton direction is summarized in Algorithm 15. In the next subsection, we show that Algorithm 15 scales linearly with respect to the number of joints.

5.9.2.4. Complexity Analysis. In Table 5.6, we present a short summary of the computational complexities for each step to compute the Gauss-Newton direction, and in Tables 5.7 and 5.8, we present a comprehensive analysis of the computational complexities that leads to results in Table 5.6. The analysis also proves the complexity conclusions in Proposition 2.

Table 5.6. The summary of the computational complexities for the steps to compute the Gauss-Newton direction for the dense unconstrained and sparse constrained formulations, where K is the number of joints, P is the number of shape parameters, N is the number of measurements for all the body parts. Note that the number of shape parameters P is assumed to be varying in (a) and constant in (b).

	Dense Unconstrained Formulation	**Sparse Constrained Formulation**
Step 1	$O\big(N(6+3K+P)\big)$	$O\big(K(9+P)\big)+O\big(N(9+P)\big)$
Step 2	$O\big(N(6+3K+P)^2\big)$	$O\big(N(9+P)^2\big)$
Step 3	$O\big((6+3K+P)^3\big)$	$O\big(K(9+P)^2\big)+O\big((6+P)^3\big)$
Total	$O\big((6+3K+P)^3\big)+O\big(N(6+3K+P)^2\big)$	$O\big(K(9+P)^2\big)+O\big((6+P)^3\big)+O\big(N(9+P)^2\big)$

(a)

	Dense Unconstrained Formulation	**Sparse Constrained Formulation**
Step 1	$O\big(KN\big)$	$O(K)+O(N)$
Step 2	$O\big(K^2N\big)$	$O\big(N\big)$
Step 3	$O(K^3)$	$O(K)$
Total	$O(K^3)+O\big(K^2N\big)$	$O(K)+O\big(N\big)$

(b)

In Tables 5.6 to 5.8, it can be concluded that our sparse constrained formulation is $O(K)$ times faster for Step 1, and $O(K^2)$ times for Steps 2 and 3 than the dense unconstrained formulation in terms of the number of joints K. In total, our sparse constrained formulation scales linearly with respect to the number of joints instead of cubically as the dense unconstrained formulation.

Furthermore, in terms of the number of measurements N, Tables 5.6 to 5.8 indicate that the complexity of our sparse constrained formulation is $O\big(N(9+P)^2\big)$ or $O(N)$, whereas that of the dense constrained formulation is $O\big(N(6+3K+P)^2\big)$ or $O(K^2N)$. This suggests that our sparse constrained formulation has the the number of joints K and measurements N decoupled in the computation, and as a result, is much more efficient to handle optimization problems with more measurements. Note that it is common in [**11**, **12**,**137**–**139**,**152**] to introduce extra measurements to improve the estimation accuracy.

Table 5.7. The analysis of the computational complexities for the steps to compute the Gauss-Newton direction for the dense unconstrained. In this table, K is the number of joints, P is the number of shape parameters, N is the number of measurements for all the body parts, and N_i is the number of measurements associated with body part i.

Step 1	(a) It takes $O\big(N_i(6+3K+P)\big)$ time to compute $\mathbf{J}_i \in \mathbb{R}^{N_i\times(6+3K+P)}$ in Eq. (5.34) for each $i = 0, \cdots, K$. (b) In total, it takes $O\big(N(6+3K+P)\big)$ time to compute $\mathbf{J}_i \in \mathbb{R}^{N_i\times(6+3K+P)}$ for all $i = 0, \cdots, K$.
Step 2	(a) It takes $O\big(N_i(6+3K+P)^2\big)$ to compute $\mathbf{J}_i^\top\mathbf{J}_i \in \mathbb{R}^{(6+3K+P)\times(6+3K+P)}$ for each $i = 0, \cdots, K$. (b) In total, it takes $O\big(N(6+3K+P)^2\big)$ time to compute $\mathbf{H} = \sum_{i=0}^{K}\mathbf{J}_i^\top\mathbf{J}_i \in \mathbb{R}^{(6+3K+P)\times(6+3K+P)}$ in Eq. (5.35).
Step 3	(a) In total, it takes $O\big((6+3K+P)^3\big)$ to compute the matrix inverse of $\mathbf{H} \in \mathbb{R}^{(6+3P+K)\times(6+3P+K)}$ and solve Eq. (5.36).
Total	The overall complexity is $O\big((6+3K+P)^3\big) + O\big(N(6+3K+P)^2\big)$.

Table 5.8. The analysis of the computational complexities for the steps to compute the Gauss-Newton direction for the dense unconstrained. In this table, K is the number of joints, P is the number of shape parameters, N is the number of measurements for all the body parts, and N_i is the number of measurements associated with body part i.

Step 1	(a) It takes $O\big(9+P\big)$ time to compute $\mathbf{A}_i \in \mathbb{R}^{(6+P)\times(6+P)}$ and $\mathbf{B}_i \in \mathbb{R}^{(6+P)\times 3}$ in Eqs. (5.41) and (5.42) for each $i = 0, \cdots, K$. Note that the bottom of $\mathbf{A}_i$ and $\mathbf{B}_i$ in Eqs. (5.41) and (5.42) are either zero or identity matrices, which simplifies the computation. (b) It takes $O\big(N_i(9+P)\big)$ time to compute $\mathbf{J}_{i,1} \in \mathbb{R}^{N_i\times(9+P)}$ and $\mathbf{J}_{i,2} \in \mathbb{R}^{N_i\times 3}$ in Eqs. (5.39) and (5.40) for each $i = 0, \cdots, K$. (c) Note that $\mathbf{J}_{i,1}$, $\mathbf{J}_{i,2}$, $\mathbf{A}_i$ and $\mathbf{B}_i$ are intermediates to compute $\mathbf{J}_i$ in Eq. (5.34) using the chain rule. (d) In total, it takes $O\big(K(9+P)+O\big(N(9+P)\big)\big)$ time to compute $\mathbf{J}_{i,1}$, $\mathbf{J}_{i,2}$, $\mathbf{A}_i$ and $\mathbf{B}_i$ for all $i = 0, \cdots, K$.
Step 2	(a) It takes $O\left(N_i(9+P)^2\right)$ time to compute $\mathbf{H}_{i,11} \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{H}_{i,21} \in \mathbb{R}^{3\times(6+P)}$ and $\mathbf{H}_{i,22} \in \mathbb{R}^{3\times 3}$ in Eq. (5.43) for each $i = 0, \cdots, K$. (b) In total, it takes $O\big(N(9+P)^2\big)$ time to compute $\mathbf{H}_{i,11} \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{H}_{i,21} \in \mathbb{R}^{3\times(6+P)}$ and $\mathbf{H}_{i,22} \in \mathbb{R}^{3\times 3}$ for all $i = 0, \cdots, K$.
Step 3	(a) It takes $O\big((9+P)^2\big)$ time to run lines 4-19 and lines 28-29 in Algorithm 15 for each $i = 1, \cdots, K$. Note that $\mathbf{A}_i$ and $\mathbf{B}_i$ in Eqs. (5.41) and (5.42) are zero and identity matrices at the bottom, which can be exploited to simplify the computation. (b) It takes $O\big((6+P)^3\big)$ time to compute the matrix inverse of $\mathbf{M}_0 \in \mathbb{R}^{(6+P)\times(6+P)}$ in line 25 of Algorithm 15. (c) In total, it takes $O\big(K(9+P)^2\big)+O\big((6+P)^3\big)$ to compute the Gauss-Newton direction.
Total	The overall complexity is $O\big(K(9+P)^2\big)+O\big((6+P)^3\big)+O\big(N(9+P)^2\big)$.

References

[1] T. Fan and T. Murphey, "Majorization minimization methods for distributed pose graph optimization with convergence guarantees," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[2] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis, "Learning to reconstruct 3d human pose and shape via model-fitting in the loop," in *Proceedings of the IEEE/CVF Conference on Computer Vision*, 2019.

[3] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," *arXiv preprint arXiv:2011.08036*, 2020.

[4] J. Lee, C. K. Liu, F. C. Park, and S. S. Srinivasa, "A linear-time variational integrator for multibody systems," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[5] 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," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1286–1311, 2017.

[6] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, "Distributed certifiably correct pose-graph optimization," *arXiv preprint arXiv:1911.03721*, 2019.

[7] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special euclidean group," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 95–125, 2019.

[8] 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 International Conference on Robotics and Automation (ICRA)*, 2015.

[9] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *2018 IEEE*

International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 2916–2923.

[10] T. von Marcard, R. Henschel, M. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3D human pose in the wild using IMUs and a moving camera," in *European Conference on Computer Vision (ECCV)*, sep 2018.

[11] D. Xiang, H. Joo, and Y. Sheikh, "Monocular total capture: Posing face, body, and hands in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 965–10 974. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Xiang_Monocular_Total_Capture_Posing_Face_Body_and_Hands_in_the_CVPR_2019_paper.html

[12] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image," in *European conference on computer vision (ECCV)*, 2016.

[13] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2013.

[14] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3D human pose estimation in the wild using improved cnn supervision," in *Proceedings of the International Conference on 3D Vision*. IEEE, 2017. [Online]. Available: http://gvv.mpi-inf.mpg.de/3dhp_dataset

[15] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.

[16] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[17] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, 2010.

[18] D. M. Rosen, K. J. Doherty, A. Terán Espinoza, and J. J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 215–242, 2021.

[19] E. R. Johnson and T. D. Murphey, "Scalable variational integrators for constrained mechanical systems in generalized coordinates," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1249–1261, 2009.

[20] M. Kobilarov, K. Crane, and M. Desbrun, "Lie group integrators for animation and control of vehicles," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 2, p. 16, 2009.

[21] E. Johnson, J. Schultz, and T. Murphey, "Structured linearization of discrete mechanical systems for analysis and optimal control," *IEEE Transactions on Automation Science and Engineering*, 2015.

[22] T. Fan and T. Murphey, "Structured linearization of discrete mechanical systems on lie groups: A synthesis of analysis and control," in *IEEE Conference on Decision and Control (CDC)*, 2015, pp. 1092–1099.

[23] O. Junge, J. E. Marsden, and S. Ober-Blöbaum, "Discrete mechanics and optimal control," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 538–543, 2005.

[24] C. Lacoursiere, "Ghosts and machines: regularized variational methods for interactive simulations of multibodies with dry frictional contacts," Ph.D. dissertation, Datavetenskap, 2007.

[25] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *International Symposium on Robotics Research (ISRR)*, 2017.

[26] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[27] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for hzd gait optimization," *IEEE Transactions on Robotics*, 2018.

[28] T. Fan, J. Schultz, and T. D. Murphey, "Efficient computation of variational integrators in robotic simulation and trajectory optimization," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, submitted, 2018.

[29] S. Ober-Blöbaum and N. Saake, "Construction and analysis of higher order Galerkin variational integrators," *Advances in Computational Mathematics*, vol. 41, no. 6, pp. 955–986, 2015.

[30] S. Ober-Blöbaum, "Galerkin variational integrators and modified symplectic Runge–Kutta methods," *IMA Journal of Numerical Analysis*, vol. 37, no. 1, pp. 375–406, 2017.

[31] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[32] K. Yamane and Y. Nakamura, "Efficient parallel dynamics computation of human figures," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2002.

[33] A. Fijany, I. Sharf, and G. M. D'Eleuterio, "Parallel O (log n) algorithms for computation of manipulator forward dynamics," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 389–400, 1995.

[34] J. Carpentier, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and Systems (RSS)*, 2018.

[35] J. Pratt and G. Pratt, "Intuitive control of a planar bipedal walking robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1998.

[36] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the LittleDog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 192–215, 2011.

[37] G. Nelson, A. Saunders, N. Neville, B. Swilling, J. Bondaryk, D. Billings, C. Lee, R. Playter, and M. Raibert, "Petman: A humanoid robot for testing chemical protective clothing," *Journal of the Robotics Society of Japan*, vol. 30, no. 4, pp. 372–377, 2012.

[38] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[39] T. Fan and T. Murphey, "Online feedback control for input-saturated robotic systems on lie groups," in *Robotics: Science and Systems Conference (RSS)*, 2016.

[40] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference (ACC)*, 2005, pp. 300–306.

[41] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1168–1175.

[42] T. Fan, H. Weng, and T. Murphey, "Decentralized and recursive identification for cooperative manipulation of unknown rigid body with local measurements," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2842–2849.

[43] J. Schultz and T. D. Murphey, "Extending filter performance through structured integration," in *American Control Conference (ACC), 2014*. IEEE, 2014.

[44] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, 2016.

[45] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[46] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[47] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[48] A. Kleiner, J. Prediger, and B. Nebel, "Rfid technology-based exploration and slam for search and rescue," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 4054–4059.

[49] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: Recent advances and new challenges," in *IFAC Conference of Manoeuvering and Control of Marine Craft*, vol. 88, 2006, pp. 1–12.

[50] J. Dong, J. G. Burnham, B. Boots, G. Rains, and F. Dellaert, "4d crop monitoring: Spatio-temporal reconstruction for agriculture," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[51] O. Vysotska and C. Stachniss, "Exploiting building information from publicly available maps in graph-based SLAM," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4511–4516.

[52] J. Polvi, T. Taketomi, G. Yamamoto, A. Dey, C. Sandor, and H. Kato, "SlidAR: A 3d positioning method for SLAM-based handheld augmented reality," *Computers & Graphics*, vol. 55, pp. 33–43, 2016.

[53] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[54] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[55] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287–300, 2002.

[56] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, 2005.

[57] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[58] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent." in *Robotics: Science and Systems*, vol. 3, 2007, p. 9.

[59] T. Fan and T. Murphey, "Generalized proximal methods for pose graph optimization," in *International Symposium on Robotics Research (ISRR)*, 2019.

[60] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[61] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[62] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*.

[63] D. M. Rosen, M. Kaess, and J. J. Leonard, "Rise: An incremental trust-region method for robust online sparse least-squares estimation," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1091–1108, 2014.

[64] S. Huang, Y. Lai, U. Frese, and G. Dissanayake, "How far is slam from a linear least squares problem?" in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 3011–3016.

[65] H. Wang, G. Hu, S. Huang, and G. Dissanayake, "On the structure of nonlinearities in pose graph slam," *Robotics: Science and Systems VIII*, p. 425, 2013.

[66] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^{2}o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*.

[67] L. Carlone and A. Censi, "From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 475–492, 2014.

[68] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 965–987, 2014.

[69] K. Khosoussi, S. Huang, and G. Dissanyake, "Exploiting the separable structure of slam," in *Robotics: Science and systems*, 2015.

[70] M. Liu, S. Huang, G. Dissanayake, and H. Wang, "A convex optimization based approach for pose slam problems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[71] L. Carlone and F. Dellaert, "Duality-based verification techniques for 2D SLAM," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[72] L. Carlone, G. C. Calafiore, C. Tommolillo, and F. Dellaert, "Planar pose graph optimization: Duality, optimal solutions, and verification," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 545–565, 2016.

[73] 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 International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[74] J. Briales and J. Gonzalez-Jimenez, "Fast global optimality verification in 3d slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[75] ——, "Cartan-sync: Fast and global se (d)-synchronization," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2127–2134, 2017.

[76] J. G. Mangelson, J. Liu, R. M. Eustice, and R. Vasudevan, "Guaranteed globally optimal planar pose graph and landmark slam via sparse-bounded sums-of-squares programming," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9306–9312.

[77] A. Singer, "Angular synchronization by eigenvectors and semidefinite programming," *Applied and computational harmonic analysis*, vol. 30, no. 1, pp. 20–36, 2011.

[78] A. Singer and Y. Shkolnisky, "Three-dimensional structure determination from common lines in cryo-em by eigenvectors and semidefinite programming," *SIAM journal on imaging sciences*, vol. 4, no. 2, pp. 543–572, 2011.

[79] A. S. Bandeira, N. Boumal, and A. Singer, "Tightness of the maximum likelihood semidefinite relaxation for angular synchronization," *Mathematical Programming*, vol. 163, no. 1-2, pp. 145–167, 2017.

[80] N. Boumal, "Nonconvex phase synchronization," *SIAM Journal on Optimization*, vol. 26, no. 4, pp. 2355–2377, 2016.

[81] A. Eriksson, C. Olsson, F. Kahl, and T.-J. Chin, "Rotation averaging and strong duality," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 127–135.

[82] M. Betke and L. Gurvits, "Mobile robot localization using landmarks," *IEEE transactions on robotics and automation*, vol. 13, no. 2, pp. 251–263, 1997.

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

[84] N. Boumal, V. Voroninski, and A. Bandeira, "The non-convex Burer-Monteiro approach works on smooth semidefinite programs," in *Advances in Neural Information Processing Systems*, 2016.

[85] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011, vol. 2.

[86] J. M. Selig, *Geometric fundamentals of robotics*. Springer Science & Business Media, 2004.

[87] T. Fan, H. Wang, M. Rubenstein, and T. Murphey, "Efficient and guaranteed planar pose graph optimization using the complex number representation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1904–1911.

[88] ——, "CPL-SLAM: Efficient and certifiably correct planar graph-based slam using the complex number representation," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1719–1737, 2020.

[89] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[90] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[91] P.-A. Absil and K. A. Gallivan, "Joint diagonalization on the oblique manifold for independent component analysis," in *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, 2006.

[92] C. Khatri and K. Mardia, "The von Mises-Fisher matrix distribution in orientation statistics," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 95–106, 1977.

[93] L. Carlone, "A convergence analysis for pose graph optimization via Gauss-Newton methods," in *2013 IEEE International Conference on Robotics and Automation*, 2013.

[94] J. Gallier, "The schur complement and symmetric positive semidefinite (and definite) matrices," 2010.

[95] C. D. Meyer, *Matrix analysis and applied linear algebra*. SIAM, 2000, vol. 71.

[96] 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.

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

[98] D. Rosen and L. Carlone, "Computational enhancements for certifiably correct slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[99] Y. Latif, C. Cadena, and J. Neira, "Robust graph slam back-ends: A comparative analysis."

[100] S. M. Nasiri, R. Hosseini, and H. Moradi, "A recursive least square method for 3d pose graph optimization problem," *arXiv preprint arXiv:1806.00281*, 2018.

[101] R. Tron, D. M. Rosen, and L. Carlone, "On the inclusion of determinant constraints in Lagrangian duality for 3d slam," *Robot. Sci. Syst. Work.âĂIJThe Probl. Mob. Sensors*, 2015.

[102] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, "Complementarity and nondegeneracy in semidefinite programming," *Mathematical programming*, 1997.

[103] D. M. Rosen, "Scalable low-rank semidefinite programming for certifiably correct machine perception," in *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, vol. 3, 2020.

[104] G. Grisetti, C. Stachniss, and W. Burgard, "Nonlinear constraint network optimization for efficient map learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009.

[105] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell, "Coordinated multi-robot planning while preserving individual privacy," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2188–2194.

[106] Y. Zhang and D. A. Shell, "Complete characterization of a class of privacy-preserving tracking problems," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 299–315, 2019.

[107] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using constrained factor graphs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[108] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues, "Multi-agent localization from noisy relative pose measurements," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 364–369.

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

[110] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[111] 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 International Conference on Robotics and Automation (ICRA)*, 2015.

[112] 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*, vol. 5, no. 2, pp. 1656–1663, 2020.

[113] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[114] 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.

[115] Y. Chang, Y. Tian, J. P. How, and L. Carlone, "Kimera-multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping," *arXiv preprint arXiv:2011.04087*, 2020.

[116] 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," *arXiv preprint arXiv:2106.14386*, 2021.

[117] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004.

[118] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2016.

[119] T. Fan and T. Murphey, "Majorization minimization methods for distributed pose graph optimization," *arXiv preprint arXiv:2108.00083*, 2021.

[120] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2)," in *Doklady AN USSR*, vol. 269, 1983, pp. 543–547.

[121] ——, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013, vol. 87.

[122] B. OâĂŹdonoghue and E. Candes, "Adaptive restart for accelerated gradient schemes," *Foundations of computational mathematics*, vol. 15, no. 3, pp. 715–732, 2015.

[123] R. Tron and R. Vidal, "Distributed 3-d localization of camera sensor networks from 2-d image measurements," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3325–3340, 2014.

[124] E. Cristofalo, E. Montijano, and M. Schwager, "Geod: Consensus-based geodesic distributed pose graph optimization," *arXiv preprint arXiv:2010.00156*, 2020.

[125] 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.

[126] 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.

[127] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 62–69.

[128] L. Carlone and G. C. Calafiore, "Convex relaxations for pose graph optimization with outliers," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1160–1167, 2018.

[129] J. T. Barron, "A general and adaptive robust loss function," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[130] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 4, pp. 376–380, 1991.

[131] A. McAdams, A. Selle, R. Tamstorf, J. Teran, and E. Sifakis, "Computing the singular value decomposition of 3x3 matrices with minimal branching and elementary floating point operations," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2011.

[132] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," *Mathematical Programming*, vol. 156, no. 1-2, pp. 59–99, 2016.

[133] C. Jin, P. Netrapalli, and M. I. Jordan, "Accelerated gradient descent escapes saddle points faster than gradient descent," in *Conference On Learning Theory*, 2018.

[134] H. Li and Z. Lin, "Accelerated proximal gradient methods for nonconvex programming," in *Advances in neural information processing systems*, 2015, pp. 379–387.

[135] H. Zhang and W. W. Hager, "A nonmonotone line search technique and its application to unconstrained optimization," *SIAM Journal on Optimization*, vol. 14, no. 4, pp. 1043–1056, 2004.

[136] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

[137] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt, "XNect: Real-time multi-person 3D motion capture with a single RGB camera," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 82–1, 2020.

[138] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "VNect: Real-time 3D human pose estimation with a single RGB camera," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.

[139] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. J. Black, and P. V. Gehler, "Unite the People: Closing the loop between 3D and 2D human representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[140] M. Loper, N. Mahmood, and M. J. Black, "MoSh: Motion and shape capture from sparse markers," *ACM Transactions on Graphics (TOG)*, 2014.

[141] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7122–7131.

[142] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[143] T. Fan, K. V. Alwala, D. Xiang, W. Xu, T. Murphey, and M. Mukadam, "Revitalizing optimization for 3d human pose and shape estimation: A Sparse Constrained Formulation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 457–11 466.

[144] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 248:1–248:16, Oct. 2015.

[145] N. Kolotouros, G. Pavlakos, and K. Daniilidis, "Convolutional mesh regression for single-image human shape reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[146] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: Shape completion and animation of people," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 408–416, Jul. 2005.

[147] A. A. A. Osman, T. Bolkart, and M. J. Black, "STAR: A spare trained articulated human body regressor," in *European Conference on Computer Vision (ECCV)*, 2020.

[148] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2019.

[149] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017, pp. 2334–2343.

[150] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5693–5703.

[151] Y. Huang, F. Bogo, C. Lassner, A. Kanazawa, P. V. Gehler, J. Romero, I. Akhter, and M. J. Black, "Towards accurate markerless human shape and pose estimation over time," in *Proceedings of the International Conference on 3D Vision*, 2017, pp. 421–430.

[152] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3D hands, face, and body from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[153] A. Zanfir, E. G. Bazavan, H. Xu, W. T. Freeman, R. Sukthankar, and C. Sminchisescu, "Weakly supervised 3d human pose and shape reconstruction with normalizing flows," in *European Conference on Computer Vision*, 2020.

[154] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Oct. 2019, pp. 5442–5451.

[155] H. Joo, T. Simon, and Y. Sheikh, "Total Capture: A 3D deformation model for tracking faces, hands, and bodies," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8320–8329.

[156] Y. Rong, T. Shiratori, and H. Joo, "Frankmocap: Fast monocular 3D hand and body motion capture by regression and integration," *arXiv preprint arXiv:2008.08324*, 2020.

[157] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3D human pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7025–7034.

[158] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3D human pose estimation in video with temporal convolutions and semi-supervised training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[159] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3D human pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017, pp. 2640–2649.

[160] G. Rogez and C. Schmid, "MoCap-guided data augmentation for 3D pose estimation in the wild," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3108–3116, 2016.

[161] G. Rogez, P. Weinzaepfel, and C. Schmid, "LCR-net: Localization-classification-regression for human pose," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[162] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[163] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[164] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D human pose estimation: New benchmark and state of the art analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014.

[165] A. Kanazawa, J. Y. Zhang, P. Felsen, and J. Malik, "Learning 3D human dynamics from video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5614–5623.

[166] A. Arnab, C. Doersch, and A. Zisserman, "Exploiting temporal context for 3D human pose estimation in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[167] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 1–17, 2017.