Distributed Pose Graph Optimization via Continuous Riemannian Dynamics

Jaeho Shin, Maani Ghaffari, and Yulun Tian
University of Michigan
Ann Arbor, MI 48109, USA
Email: {jaehos,maanigj,yulunt}@umich.edu

***Abstract*—We present a framework for distributed Pose Graph Optimization (PGO) by formulating the problem as a second-order continuous-time dynamical system evolving on Lie groups. By modeling pose variables as massive particles subject to damping, the equilibrium points of the resulting Riemannian dynamics coincide with first-order critical points of the original PGO problem. Using the governing damped Euler–Poincaré equations and a semi-implicit geometric integrator, we design an optimization algorithm that generalizes existing algorithms such as Riemannian gradient descent and Gauss–Newton. In multi-robot settings, we present a fully distributed and parallel method based on block-diagonal mass and damping matrices, where each robot solves an ordinary differential equation for its own poses with minimal communication overhead. Moreover, modeling both state and velocity enables principled neighbor prediction that significantly improves convergence under delayed communication. Theoretically, we present an analysis and establish sufficient condition that ensures energy dissipation under the employed geometric discretization scheme. Experiments on benchmark PGO datasets demonstrate that the proposed solver achieves superior performance compared to state-of-the-art distributed baselines in both synchronous and asynchronous regimes.**

I. Introduction

Multi-robot collaborative SLAM (CSLAM) is a critical capability for enabling scalable and globally consistent situational awareness in GPS-denied environments. By allowing multiple robots to jointly estimate their trajectories and build shared maps from noisy relative measurements, CSLAM significantly extends the spatial and temporal scope achievable by single-robot systems. Recent advances have led to robust, fielded multi-robot SLAM systems operating over large teams and long durations, even under intermittent communication and limited bandwidth [1–5]. At the core of these systems lies pose graph optimization (PGO), which serves as the primary back-end for fusing intra- and inter-robot measurements into a coherent global estimate.

To address scalability, communication, and privacy constraints, a growing body of recent work has investigated distributed pose graph optimization based on distributed optimization [6–8] or probabilistic message passing frameworks [9]. Compared to centralized PGO solvers, these approaches eliminate single points of failure and allow flexible deployment across robot teams. However, existing distributed PGO methods face persistent limitations in convergence speed and robustness. Generic solvers based on first-order primal (e.g., [10]) or primal–dual (e.g., [7]) iterations are broadly applicable but often converge slowly in large, loosely coupled graphs, particularly under asynchronous updates and stale neighbor information. Other state-of-the-art methods achieve acceleration by carefully exploiting problem-specific algebraic structure [8]. Frequently, this design requires substantial domain expertise and tightly couples the resulting algorithm to specific problem formulations.

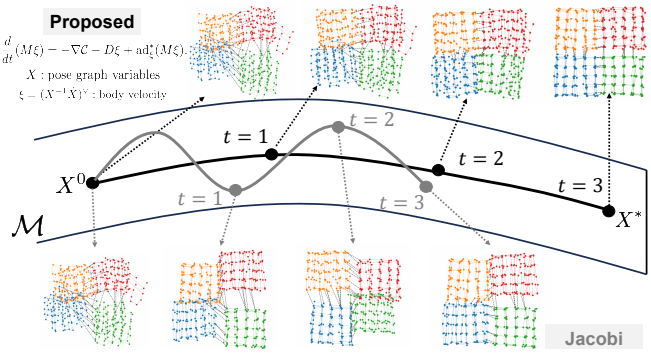

Fig. 1: The proposed approach formulates pose graph optimization (PGO) as a **continuous-time dynamical system** evolving on the direct product $\mathcal{M}$ of SE(3) Lie groups governed by a damped Euler-Poincaré equation. Using a suitable geometric integration scheme, we obtain optimization trajectory (shown in solid black) that converges to a first-order critical point of PGO. Each point along the trajectory represents a full pose graph estimate, where colors correspond to different robots. By modeling inertia and damping, the proposed approach suppresses oscillatory behavior and converges significantly faster than existing methods such as Jacobi (gray trajectory).

To address the aforementioned limitations, we propose a novel distributed PGO solver based on continuous-time Riemannian dynamics (Fig. 1), which naturally introduces inertia and neighbor state prediction to enables accelerated convergence even under asynchronous optimization. Our approach draws inspiration from the revered field of geometric mechanics [11], which has played a central role in geometric control and motion planning (e.g., [12, 13]) yet has so far remained largely unexplored in the context of SLAM back-end optimization. By studying the variational characterization of this system, we derive the governing equations of motion and show that appropriate choices of kinetic energy and geometric discretization lead to effective update rules that can be executed fully in a parallel and distributed manner.

Contributions. In summary, we present three contributions,

(i) We present **CORD** (Continuous-time Riemannian Dynamics), a general distributed PGO solver based on a second-order ordinary differential equation (ODE) evolving on Lie Groups, achieving acceleration even under delayed communication;
(ii) We give a **theoretical analysis** that precisely characterizes the energy dissipation properties of the discretized Riemannian dynamics;
(iii) **Extensive experimental results** show that CORD achieves state-of-the-art performance in synchronous PGO with both chordal and geodesic metrics, while maintaining robustness under diverse asynchronous conditions through the proposed neighbor state prediction.

II. Related Work

Distributed Pose Graph Optimization. DDF-SAM [14, 15] is a pioneering work that introduced a fully distributed SLAM back-end in which agents exchange compact summaries of their local factor graphs obtained via Gaussian elimination. DGS [16] proposes a two-stage iterative optimization procedure that uses successive over-relaxation to solve chordal rotation initialization and a single Gauss-Newton update. AMM-PGO [8] introduces a majorization–minimization framework that supports Nesterov acceleration and adaptive restart strategies in fully distributed settings. In [6], RBCD and its accelerated variant RBCD++ performs nonlinear block-coordinate descent with first-order convergence guarantees, and attains certifiable global optimality for distributed PGO when used within a semidefinite relaxation framework [17, 18]. IRBCD [19] improves convergence and scalability through enhanced graph partitioning strategies. In [10], Tian et al. develops a variant of RBCD for asynchronous optimization. The resulting ASAPP solver enjoys provable first-order convergence under bounded communication delay albeit with slower convergence speed. Recently, primal-dual iterations based on the Alternating Direction Method of Multipliers (ADMM) has gained increasing attention [7, 20, 21]. MESA [7] and its incremental variant iMESA [20] applies on-manifold consensus ADMM to distributed SLAM, and demonstrates flexible communication schedules and empirical robustness to intermittent connectivity. Gaussian belief propagation (GBP) [9] proposes a probabilistic message-passing framework that enables fully asynchronous operation and naturally provides estimates of marginal uncertainty, though the convergence behavior can be sensitive to graph topology and noise characteristics. To address this issue, recent work [22] introduces a hybrid method that integrates maximum a-posteriori optimization with belief propagation and uses Hellinger-distance-based damping to guarantee convergence on loopy graphs. In contrast to prior methods, we propose a unifying framework for distributed PGO based on continuous-time Riemannian dynamics. To the best of our knowledge, our approach is the first to achieve acceleration even under asynchronous communication.

Dynamical System Perspective in Optimization. Su et al. [23] is among the first to study Nesterov's accelerated gradient method [24] using a continuous-time second-order ODE. Subsequent work [25, 26] generalize this perspective by introducing a Bregman Lagrangian whose Euler–Lagrange equations define a family of continuous-time dynamics, and show that appropriate discretizations of these dynamics recover existing accelerated optimization methods. Recent methods [27, 28] further place this framework in a Hamiltonian setting and show that symplectic integration [29] is essential for preserving acceleration when passing from continuous to discrete time. Parallel efforts [30, 31] study the continuous-time dynamics perspective for optimization over Riemannian manifolds.

In robotics, [32–37] study the related problem of point cloud registration from the dynamical system perspective. Golyanik et al. [32] formulate registration as a damped N-body dynamical system, where template points evolve under gravitational forces induced by the reference points. This approach is extended in [34] that introduces a nonlinear least squares optimization over an altered potential and uses Barnes–Hut tree to accelerate point interaction. Yang et al. [37] generalizes [32–34] and models a broader range of pose estimation problems across point, primitive, and category-level settings. This work extends dynamical systems–based optimization to distributed pose graph optimization, a significantly higher-dimensional and poorly-conditioned estimation problem, and explicitly accounts for communication delays that fundamentally challenge existing distributed solvers.

III. Problem Formulation

We use $\mathcal{G}$ and $\mathfrak{g}$ to denote a matrix Lie group and its associated Lie algebra. In particular, we use $\mathrm{SE}(3)$ to denote the special Euclidean group in three dimensions, and $\mathrm{SE}(3)^n$ to denote the direct product of n copies of $\mathrm{SE}(3)$.

Let $X \equiv X(t) \in \mathcal{G}$ denote a trajectory evolving on $\mathcal{G}$. We formulate the dynamics using the body velocity ξ, defined by left-translating the time derivative $\dot{X}$ to the Lie algebra $\mathfrak{g}$ associated with $\mathcal{G}$:

$$\xi := X^{-1}\dot{X} \in \mathfrak{g}. \tag{1}$$

In classical mechanics, the evolution of a system is described using a *Lagrangian* $\mathcal{L} = \mathcal{T} - \mathcal{C}$, defined as the difference between kinetic energy $\mathcal{T}$ and potential energy $\mathcal{C}$. When the configuration space is a Lie group, this formulation extends naturally to equations of motion that evolve directly on the manifold and respect its geometric structure [11]. This perspective serves as the foundation of the proposed method introduced in later sections.

A. Pose Graph Optimization (PGO)

A pose graph is represented by a directed graph $G = (\mathcal{V}, \mathcal{E})$, where each node $u \in \mathcal{V}$ corresponds to an unknown robot pose $X_u \in \mathrm{SE}(3)$, and each edge $(u, v) \in \mathcal{E}$ encodes a noisy relative pose measurement $X_{uv} \in \mathrm{SE}(3)$ between nodes u and v. The objective of PGO is to estimate the set of poses

$\{X_u\}_{u\in\mathcal{V}}$ that best agree with the relative measurements. In this paper, we study two widely used formulations of PGO corresponding to geodesic and chordal distance metrics on $\mathrm{SE}(3)$, which together cover the objective functions underlying state-of-the-art solvers such as GTSAM [38] and SE-Sync [17].

Problem 1 (Pose Graph Optimization)**.**

$$\min_{X\in\mathrm{SE}(3)^N} \quad \mathcal{C}(X) \coloneqq \sum_{(u,v)\in\mathcal{E}} \|r_{uv}(X_u, X_v)\|^2_{\Omega_{uv}},$$
$$r_{uv}(X_u, X_v) = \begin{cases} \log\left(X_{uv}^{-1} X_u^{-1} X_v\right), & \textit{(Geodesic)}, \\ X_u X_{uv} - X_v, & \textit{(Chordal)}. \end{cases} \tag{2}$$

Here, $X \in \mathrm{SE}(3)^N$ collects all $N \coloneqq |\mathcal{V}|$ pose variables and $\Omega_{uv} \succ 0$ denotes the information matrix associated with edge $(u, v) \in \mathcal{E}$. The weighted squared norm is defined as $\|r_{uv}\|^2_{\Omega_{uv}} \coloneqq r_{uv}^\top \Omega_{uv} r_{uv}$ for the case of the geodesic residual. For the chordal residual, r_{uv} is matrix-valued and the weighted norm is defined as $\|r_{uv}\|^2_{\Omega_{uv}} \coloneqq \mathrm{tr}\left(r_{uv}\,\Omega_{uv}\,r_{uv}^\top\right)$, where $\Omega_{uv} = \mathrm{blkdiag}(\omega_{R_{uv}} I_3,\ \omega_{t_{uv}})$ weights the rotational and translational components [17, 18].

In distributed PGO, the pose graph is naturally partitioned across robots, with each robot estimating a subset of pose variables corresponding to its own trajectory. Edges include both intra-robot measurements (e.g., odometry and local loop closures) and inter-robot measurements arising from loop closures established when different robots observe the same place. The objective in distributed PGO is to solve Problem 1 collectively and without a central coordinator, using only local computation and limited information exchange between neighboring robots; see Fig. 2.

IV. Proposed Method

In this section, we develop the proposed CORD framework for distributed PGO. Sec. IV-A converts PGO to a continuous-time dynamical system evolving on the product of $\mathrm{SE}(3)$ manifolds, and show that its equilibrium states correspond to first-order critical points (FOCPs) of the PGO problem. Sec. IV-B then derives the optimization algorithm resulting from geometric semi-implicit integration. Lastly, Sec. IV-C specializes to fully distributed optimization that remains robust under asynchronous communication.

A. From PGO to Riemannian Dynamics

The majority of optimization algorithms for solving Problem 1 iteratively linearizes the residual r_{ij} and obtain a descent direction toward a FOCP. In contrast, our algorithm first formulates a continuous-time dynamical system with a potential energy based on the PGO cost $\mathcal{C}$ and a suitably chosen kinetic energy $\mathcal{T}$. Define the body velocity for each pose as $\xi_i \coloneqq (X_i^{-1}\dot{X})^\vee \in \mathbb{R}^6$ where the operator $(\cdot)^\vee$ maps a Lie algebra element to a vector in $\mathbb{R}^6$. Given a generalized inertia ("mass") matrix $M \in \mathbb{R}^{6N\times 6N}$ to be specified later, the total kinetic energy is defined as $\mathcal{T} \coloneqq \frac{1}{2}\xi^\top M\xi$, where $\xi = [\xi_1^\top, \cdots, \xi_N^\top]^\top$. Denoting the Lagrangian of the system as $\mathcal{L} \coloneqq \mathcal{T} - \mathcal{C}$, we can then derive the equation of motion for the system using the Euler-Poincaré formula [11]. Under the principle of least action, this system has equilibrium configurations corresponding to FOCPs of the potential energy $\mathcal{C}$. Nevertheless, in the absence of dissipative forces, the total energy of the system is conserved, which results in persistent oscillations rather than convergence. To ensure convergence to the FOCP, we introduce an additional positive definite damping matrix $D \succ 0$ to enable energy dissipation. The equation of motion can then be derived using the Lagrange-d'Alembert principle [11], as shown in the appendix in detail.

Proposition 1 (Damped Euler-Poincaré Equation)**.** *Given positive definite mass matrix M and damping matrix D, the system dynamics on $\mathrm{SE}(3)^N$ are governed by,*

$$\frac{d}{dt}(M\xi) = -\nabla\mathcal{C} - D\xi + \mathrm{ad}^*_\xi(M\xi). \tag{3}$$

In (3), ad^*_ξ denotes the *co-adjoint operator* associated with the Lie algebra element ξ, defined via $\langle \mathrm{ad}^*_\xi\,\mu, \eta\rangle = \langle \mu, \mathrm{ad}_\xi\,\eta\rangle = \langle \mu, [\xi, \eta]\rangle$ where $[\cdot,\cdot]$ denotes the Lie bracket on $\mathfrak{g}$. Intuitively, this term in (3) captures curvature-induced coupling between velocity and momentum that arises when dynamics evolve on a Lie group (e.g., $\mathrm{SE}(3)$), and it vanishes in Euclidean spaces.

In the proposed method, we allow M to be either constant or state-dependent, i.e., $M \equiv M(X)$. In the latter case, the left-hand side time derivative in (3) expands as $\frac{d}{dt}\left(M\dot{\xi}\right) = M\dot{\xi} + \dot{M}\xi$. Moving the second term $\dot{M}\xi$ to the right-hand side of (3),

$$M\dot{\xi} = -\nabla\mathcal{C} - D\xi + \mathrm{ad}^*_\xi(M\xi) - \dot{M}\xi. \tag{4}$$

Intuitively, the resulting dynamics includes a corresponding correction term to account for the varying value of M. When M is constant, the $\dot{M}\xi$ term also vanishes.

Remark 1. *We note that any steady-state solution of* (3) *satisfies the FOCP condition of Problem 1: at steady state, $\dot{\xi} = \xi = 0$ so that* (3) *reduces to $\nabla\mathcal{C} = 0$.*

B. Optimization via Geometric Discretization

To numerically integrate the continuous-time dynamics in (3), we use a geometric semi-implicit Euler scheme as shown in Alg. 1. At each iteration, we first evaluate the induced forces according to Prop. 1 (lines 5–8). Specifically, line 5 evaluates the potential-induced force $F_{\mathrm{grad}} = -\nabla\mathcal{C}(X_k)$ that drives the state toward a FOCP. Line 6 computes the damping term needed for convergence, and line 7 corresponds to the co-adjoint term in (3) that accounts for the Lie-group geometry in the momentum dynamics. When the mass matrix is state-dependent, the additional term F_{varM} in line 8 serves as a discrete approximation of the $\dot{M}\xi$ term in (4). Given the computed forces, lines 10–12 then performs a forward Euler step on the velocity. In lines 14, we use a semi-implicit scheme so that the update uses the *new* velocity ξ_{k+1} rather than ξ_k. This choice significantly improves stability over a fully explicit update $X_{k+1} = X_k \exp((\xi_k \Delta t)^\wedge)$ while avoiding the nonlinear solves required by fully implicit integrators. Finally,

Algorithm 1 Riemannian ODE Solver

Require: Initial state $(X_0^i, \xi_0^i) \in \{SE(3) \times \mathbb{R}^6\}^N$, Initial mass $M_0 \in \mathbb{R}^{6N\times 6N}$, Damping $D \in \mathbb{R}^{6N\times 6N}$, Step size Δt

1: Initialize body velocity $\boldsymbol{\xi}_0 \leftarrow \mathbf{0}_{6N}$
2: $k \leftarrow 0$
3: **while** $k <$ MaxIter **do**
4: **1. Compute Forces:**
5: $F_{\text{grad}} \leftarrow -\nabla\mathcal{C}(X_k)$
6: $F_{\text{damp}} \leftarrow -D\xi_k$
7: $F_{\text{cor}} \leftarrow \text{ad}^*_{\xi_k}(M_k\xi_k)$
8: $F_{\text{varM}} \leftarrow -\left(\frac{M_k - M_{k-1}}{\Delta t}\right)\xi_k$
9: **2. Numerical Integration:**
10: $F_{\text{total}} \leftarrow F_{\text{grad}} + F_{\text{damp}} + F_{\text{cor}} + F_{\text{varM}}$
11: $a_k \leftarrow M_k^{-1} F_{\text{total}}$
12: $\xi_{k+1} \leftarrow \xi_k + a_k\Delta t$
13: **3. Manifold Update:**
14: $X_{k+1} \leftarrow X_k \exp((\xi_{k+1}\Delta t)^\wedge)$
15: Update Mass Matrix M_{k+1} based on X_{k+1}
16: $k \leftarrow k+1$
17: **end while**
18: **return** Optimized pose X_k

if M depends on the current state, we refresh M_{k+1} using the updated pose (line 15).

Before proceeding, we present a connection between the proposed second-order dynamics and existing optimization algorithms via an overdamped limiting argument. Specifically, consider scaling the mass matrix as $M = \varepsilon\widetilde{M}$ with $\varepsilon > 0$ and taking the limit $\varepsilon \to 0$. In this regime, the inertial term vanishes and the body velocity becomes $D\xi = -\nabla\mathcal{C}$, yielding a first-order gradient flow on the manifold. Under an explicit time discretization with step size Δt, this induces the update $X_{k+1} = X_k \exp(\Delta t\, \xi_k^\wedge)$. Choosing $D = I$ recovers Riemannian gradient descent [39]. Alternatively, choosing the damping matrix as $D = J^\top\Omega J$ yields the Gauss–Newton direction $\xi = -(J^\top\Omega J)^{-1}\nabla\mathcal{C}$, where J denotes the Jacobian of the stacked residuals with respect to the pose variables and Ω is the block-diagonal information matrix collecting Ω_{ij} for all edges [38]. Viewed from this perspective, gradient descent and Gauss–Newton arise as *first-order and overdamped limits* of the proposed second-order Riemannian dynamics.

C. Distributed Riemannian Dynamics for Multi-Robot PGO

In this section, we specialize our formulation to distributed PGO. Specifically, by choosing the mass and damping matrices to be *block-diagonal* with respect to a robot-wise partition of the pose graph, the resulting dynamics naturally decompose across robots while preserving the original global objective. Concretely, we select $M = \text{blkdiag}(M^1, \dots, M^R)$ and $D = \text{blkdiag}(D^1, \dots, D^R)$, where $M^i, D^i \in \mathbb{R}^{6n_i\times 6n_i}$ correspond to robot i and n_i is the number of poses owned by that robot. Under this choice, the inertial, damping, and co-adjoint terms in (3) are entirely local, yielding the decomposed Riemannian dynamics for each robot i,

$$\frac{d}{dt}(M^i\xi^i) = -\nabla\mathcal{C}^i - D^i\xi^i + \text{ad}^*_{\xi^i}(M^i\xi^i). \tag{5}$$

In (5), $\nabla\mathcal{C}^i$ denotes the block of the Riemannian gradient that corresponds to robot i's variables. For robot i to compute $\nabla\mathcal{C}^i$, note that the portion of the global PGO cost involving its pose variables can be written as

$$\mathcal{C}^i = \underbrace{\sum_{(u,v)\in\mathcal{E}_i^{\text{intra}}} \|r_{uv}\|^2_{\Omega_{uv}}}_{\text{Intra-robot}} + \underbrace{\sum_{j\in\mathcal{N}_i}\sum_{(u,v)\in\mathcal{E}_{ij}^{\text{inter}}} \|r_{uv}(X_u, \bar{X}_v)\|^2_{\Omega_{uv}}}_{\text{Inter-robot}}, \tag{6}$$

where $\mathcal{E}_i^{\text{intra}}$ denotes edges connecting poses owned by robot i, $\mathcal{E}_{ij}^{\text{inter}}$ denotes edges between robot i and its neighbor j, and $\bar{X}_v$ denotes the pose of a neighbor robot (treated as fixed). The Riemannian gradient block $\nabla\mathcal{C}^i$ in (5) is obtained by differentiating (6) with respect to robot i's pose variables. The intra-robot contribution depends only on locally available states, while the inter-robot contribution requires pose information from neighboring robots connected by inter-robot edges. As a result, $\nabla\mathcal{C}^i$ can be evaluated using a single round of neighbor communication, *without* requiring global information or centralized coordination.

Choice of Mass and Damping Matrices. We design the block-diagonal mass and damping matrices to exploit second-order information of the PGO objective while preserving a fully distributed implementation. Let $H = J^\top\Omega J + \lambda I$ denote the Levenberg–Marquardt approximation of the global Hessian of $\mathcal{C}$, and let $H^i \in \mathbb{R}^{6n_i\times 6n_i}$ denote the diagonal block of H corresponding to the pose variables owned by robot i. We define the mass and damping matrices for robot i as $M^i = m\,H^i$ and $D^i = (d/t + \epsilon_d)\,H^i$, where m, d, ϵ_d are scalar coefficients. For D, the time-decaying factor d/t yields behavior analogous to accelerated gradient flows [23] while the small constant $\epsilon_d > 0$ preserves positive definiteness. With this choice, the resulting dynamics are naturally preconditioned by $(H^i)^{-1}$, aligning the gradient-induced acceleration with a local Levenberg–Marquardt direction. Similar to the gradient computation, we remark that both M^i and D^i can be formed in a distributed manner: since the block H^i depends only on second-order contributions from intra-robot edges and inter-robot edges incident to robot i, it can be assembled using local information and one round of neighbor communication.

Substituting M^i and D^i into (5), we obtain,

$$M^i\dot{\xi}^i = -\nabla\mathcal{C}^i - D^i\xi^i + \text{ad}^*_{\xi^i}(M^i\xi^i) - \dot{M}^i\xi^i, \tag{7}$$

which is the analog of (4) in the distributed setting. As before, the dynamics in (7) are integrated numerically using the geometric semi-implicit scheme. Recall that the proposed framework also allows *constant* choices of M^i and D^i. In this case, H^i is computed once at the initial estimate and held fixed afterwards, and the $\dot{M}^i\xi^i$ term in (7) becomes zero. In our ablation study (Sec. VI-D), we show that the constant setting remains competitive for PGO, thus enabling a trade-off between computational cost and adaptivity.

Unlike Gauss-Seidel-style [6, 16] schemes that update blocks (robots) sequentially, the distributed dynamics in (7) can be integrated in a fully distributed and parallel manner. Under the assumption of full synchronization, each robot updates its state using the neighbor poses in the previous round. In contrast to sequential update, which imposes sequential

Algorithm 2 Distributed Riemannian ODE Solver (Robot i)

Require: Initial state (X_0^i, ξ_0^i), Mass M^i, Damping D^i, Step size Δt.

1: $k \leftarrow 0$
2: **while** $k <$ MaxIter **do**
3: **1. Read & Predict Neighbors:**
4: **for** $j \in \mathcal{N}_i$ **do**
5: Read latest packet (X_τ^j, ξ_τ^j) with $\tau < t_k$.
6: $\hat{X}_k^j \leftarrow X_\tau^j \cdot \exp\left(\left(\xi_\tau^j (t_k - \tau)\right)^\wedge\right)$
7: **end for**
8: **2. Compute Forces:**
9: $F_{\text{grad}}^i \leftarrow -\nabla \mathcal{C}_{\text{intra}}^i(X_k^i) - \sum_{j \in \mathcal{N}_i} \nabla \mathcal{C}_{\text{inter}}^{ij}(X_k^i, \hat{X}_k^j)$
10: $F_{\text{dyn}}^i \leftarrow -D^i \xi_k^i + \text{ad}_{\xi_k^i}^*(M_k^i \xi_k^i) - \left(\frac{M_k^i - M_{k-1}^i}{\Delta t}\right) \xi_k^i$
11: **3. Semi-Implicit Integration:**
12: $\xi_{k+1}^i \leftarrow \xi_k^i + (M_k^i)^{-1}(F_{\text{grad}}^i + F_{\text{dyn}}^i)\Delta t$
13: $X_{k+1}^i \leftarrow X_k^i \cdot \exp\left((\xi_{k+1}^i \Delta t)^\wedge\right)$
14: $k \leftarrow k + 1$
15: **end while**

dependencies that often force agents to wait, our approach eliminates such idle time ensuring that all robots compute their updates simultaneously. This independence simplifies the system architecture, and allows communication and optimization to be implemented intuitively. Moreover, the inertia and damping built in our formulation helps avoid the instability of standard Jacobi-style [16] updates.

Asynchronous Optimization via Delay Compensation. In multi-robot SLAM, communication between robots is often subject to latency, packet loss, and asynchronous message arrival, making it impractical to assume access to up-to-date neighbor states at every iteration. Thus, distributed PGO algorithms must explicitly account for delayed information when evaluating inter-robot constraints.

To this end, we note that the proposed dynamics naturally admit a principled *delay-compensation* mechanism. Alg. 2 presents our algorithm from the point of view of a single robot i. In addition to exchanging pose estimates, each robot also transmits its body velocity, which is cached locally upon reception. Since the system state evolves continuously on the Lie group and is driven by body velocities, a robot can predict a neighbor's current pose even when the received information is outdated. Specifically, let X and ξ denote the most recently received pose and body velocity of a neighboring robot, and let δt denote the elapsed time since the message timestamp. The predicted neighbor pose is then computed as $\hat{X} = X \exp\big((\xi\, \delta t)^\wedge\big)$, as shown in lines 3-7. Using the predicted pose, the resulting preconditioned dynamics (7) are then integrated using the geometric semi-implicit scheme (lines 8-13), similar to Sec. IV-B. This allows all robots to advance their states in parallel without requiring synchronized communication.

V. Convergence Analysis

This section presents a theoretical analysis for the general Algorithm 1 in Sec. IV. The results also extend to the distributed Algorithm 2 under synchronous communication as a special case. Our current analysis assumes the mass matrix M to be constant. In this case, we show that the proposed semi-implicit integrator admits a clear energy dissipation characterization, which leads to an explicit sufficient condition on the choice of step size that decreases the total system energy. As a result, the analysis establishes a rigorous theoretical justification for the proposed dynamics and provides practical guidance for step-size selection in both centralized and distributed implementations.

The main idea is to derive the condition for the total energy of the system $E = \mathcal{T} + \mathcal{C}$ to dissipate every time step. We present a general analysis by building on the Lipschitz-type gradient property for pullbacks introduced in [40]. Boumal et al. [40] showed that this property holds on compact matrix submanifolds under mild conditions. In [6], Tian et al. showed that the chordal distance formulation of PGO satisfies this condition within the sublevel set of the cost function.

Assumption 1 (Lipschitz-type gradient [40])**.** *Let $g_{X_k} : \mathbb{R}^{6N} \to \mathbb{R}$ be the pullback of the cost function $\mathcal{C}$ defined on the tangent space at X_k as:*

$$g_{X_k}(\eta) \coloneqq \mathcal{C}(X_k \exp(\eta^\wedge)). \tag{8}$$

$\mathcal{C}$ has Lipschitz-type gradient if the following holds for any update step $\eta \in \mathbb{R}^{6N}$,

$$g_{X_k}(\eta) \leq g_{X_k}(0) + \langle \nabla g_{X_k}(0), \eta \rangle + \frac{L}{2}\|\eta\|^2. \tag{9}$$

This property states that the cost function is locally well-behaved with the second-order terms upper bounded by the squared norm of the step on the tangent space. Given Assumption 1, we establish an upper bound on the total energy difference $\Delta E_k = E_{k+1} - E_k$ after a single step of integration.

Lemma 1 (Upper Bound on Energy Change)**.** *Under Assumption 1, the change in total energy is bounded as follows:*

$$\begin{aligned} \Delta E_k \leq & \Delta t^2 (a_k^\top \nabla \mathcal{C}_k + \frac{L}{2} ||\xi_{k+1}||_M^2 + \\ & \frac{1}{2} ||a_k||_M^2) - \Delta t \xi_k^\top D \xi_k, \end{aligned} \tag{10}$$

where $\mathcal{C}_k \coloneqq \mathcal{C}(X_k)$ and $a_k \coloneqq M^{-1}(-\nabla \mathcal{C}_k - D\xi_k + \text{ad}_{\xi_k}^*(M\xi_k))$.

The result of Lemma 1 can be separated into two terms which differ by the order of multiplied Δt. The terms involving quadratic factors represent numerical errors arising from discretization. As the time step Δt decreases, yielding more precise integration, the damping term becomes dominant. Indeed, divide both sides of (10) by Δt and taking the limit for $\Delta t \to 0$, we recover the continuous-time derivative of the total energy,

$$\dot{E} = -\xi^\top D \xi < 0, \text{ when } \xi \neq 0. \tag{11}$$

Thus, under the ideal continuous-time evolution, the energy strictly decays due to the positive definite damping D. The central challenge is to extend such energy dissipation result to the discrete-time system. Our key insight is that by appropriately selecting the step size Δt, the discretization errors (on the right-hand side of (10)) can be bounded so that the

TABLE I: Optimality gap achieved by synchronous (left) and asynchronous (right) methods after 100 iterations. Best and second best methods are shown in **bold** and underline, respectively. For CORD, we report results with both dataset-specific parameter tuning and a constant, dataset-independent parameter setting, denoted as CORD (const.).

Dataset	# Nodes	# Edges	$\mathcal{C}^{(0)}$	$\mathcal{C}^*$	$\mathcal{C}^{(100)}$						
					Synchronous				Asynchronous		
					AMM-PGO [8]	DJ	CORD	CORD (const.)	Delay = 5		Rand. Delay
									DJ	CORD	CORD (const.)
Sm. Grid	125	297	1.5614×10^3	1.0254×10^3	$\mathbf{1.0254 \times 10^3}$	1.0299×10^3	$\mathbf{1.0254 \times 10^3}$	1.0256×10^3	1.0377×10^3	$\mathbf{1.0349 \times 10^3}$	1.0313×10^3
Sphere	2500	4949	1.9709×10^3	1.6870×10^3	$\mathbf{1.6870 \times 10^3}$	1.6881×10^3	1.6872×10^3	1.6872×10^3	1.6998×10^3	$\mathbf{1.6966 \times 10^3}$	1.6886×10^3
Torus	5000	9048	2.4668×10^4	2.4227×10^4	$\mathbf{2.4227 \times 10^4}$	2.4230×10^4	$\mathbf{2.4227 \times 10^4}$	$\mathbf{2.4227 \times 10^4}$	2.4257×10^4	$\mathbf{2.4250 \times 10^4}$	2.4233×10^4
Grid	8000	22236	8.7265×10^4	8.4319×10^4	$\mathbf{8.4319 \times 10^4}$	8.4327×10^4	$\mathbf{8.4319 \times 10^4}$	$\mathbf{8.4319 \times 10^4}$	8.4356×10^4	$\mathbf{8.4346 \times 10^4}$	8.4328×10^4
Cubicle	5750	16869	8.3504×10^2	7.1713×10^2	7.1799×10^2	7.2222×10^2	$\mathbf{7.1752 \times 10^2}$	7.1756×10^2	7.2902×10^2	$\mathbf{7.2729 \times 10^2}$	7.2351×10^2
Rim	10195	29743	8.0840×10^3	5.4609×10^3	5.4817×10^3	5.6688×10^3	$\mathbf{5.4748 \times 10^3}$	$\mathbf{5.4748 \times 10^3}$	5.8761×10^3	$\mathbf{5.8424 \times 10^3}$	5.7364×10^3
Garage	1661	6275	1.4175×10^0	1.2625×10^0	1.2683×10^0	1.2761×10^0	$\mathbf{1.2655 \times 10^0}$	$\mathbf{1.2655 \times 10^0}$	1.2866×10^0	$\mathbf{1.2857 \times 10^0}$	1.2797×10^0

damping term dominates, yielding a net decrease in total energy. The following theorem formalizes this intuition by deriving a sufficient condition on Δt for which the discrete-time system preserves the energy dissipation property of the continuous-time dynamics, thereby ensuring the stability of the proposed algorithm.

Theorem 1 (Energy dissipation of Algorithm 1)**.** *Under Assumption 1 and with constant and positive definite M and D, if the time step Δt satisfies the following condition at each iteration k, then the total energy E_k is non-increasing along the iterates of Algorithm 1:*

$$\Delta t \leq \frac{\xi_k^\top D \xi_k}{\frac{1}{2}\|\nabla \mathcal{C}_k\|^2_{M^{-1}} + \frac{L}{2}\|\xi_{k+1}\|^2_M + \|a_k\|^2_M}. \tag{12}$$

Theorem 1 provides theoretical guidance for selecting a sufficiently small step size that preserves the energy dissipation property. We note that the result covers the distributed algorithm in Algorithm 2 with synchronized communication as a special case, where we use block-diagonal mass and damping matrices (M, D) that decompose the global dynamics into parallel robot-wise subsystems.

While Theorem 1 suggests the resulting dynamics converge to a FOCP, CORD is still a local optimization solver as are AMM-PGO [8] and ASAPP [10]. Global optimality guarantees could be pursued by augmenting the local search with an additional distributed verification as in DC2-PGO [6].

VI. Experiment

In this section, we evaluate the proposed method on standard PGO benchmarks under both synchronous and asynchronous communication. Our results demonstrate that the proposed Riemannian dynamics consistently achieve faster convergence and lower final cost than existing distributed baselines, and is particularly competitive for asynchronous optimization.

Experiment Setup. Throughout the evaluation, we denote our proposed approach (Algorithm 2) as CORD. Following prior work [6–8], we generate distributed PGO problem by partitioning each pose graph dataset to simulate multiple robots (by default five); see Fig. 2. Inter-robot communication is modeled using a packet-based simulation with configurable delays. At each iteration, robots exchange pose and velocity estimates with neighbors, and delayed packets are buffered to emulate asynchronous communication. The proposed approach is implemented in Python using PyTorch. Consistent with prior work [10, 41], the step size, mass, and damping parameters were determined empirically and reported in the appendix. All experiments are conducted on a workstation with Intel Core Ultra 9 285K CPU and 64GB memory.

Metrics and Baselines. For quantitative evaluation, we use GTSAM [38] and SE-Sync [17] to compute the centralized reference solution for the geodesic and chordal distance formulations, respectively. Following prior work [6, 8], we report performance by computing the relative optimality gap $(\mathcal{C}-\mathcal{C}^*)/\mathcal{C}^*$, where $\mathcal{C}$ and $\mathcal{C}^*$ denote the achieved and reference costs. For the chordal distance formulation, we select AMM-PGO [8] as our baseline as it has been shown to outperform other techniques including RBCD++ [6] and DGS [16]. For the geodesic distance formulation, we select the state-of-the-art MESA solver [7] as the baseline. We note that the original MESA implementation adopts an edge-based communication model, in which only a single pair of robots communicates at each iteration. To ensure a fair comparison with our fully parallel update scheme, we modify the MESA implementation so that all robots update their local variables at every iteration. All remaining hyperparameters for all baselines are set to the default values. Lastly, for both the chordal and geodesic distance formulations, we also include a distributed Jacobi (DJ) baseline in which each robot performs a preconditioned gradient descent update using the diagonal block of the Hessian corresponding to its local variables. For the main asynchronous experiments, we use DJ as the primary baseline, which is equivalent to the update rule used in ASAPP [6] in this setting. AMM-PGO is not included because the method is designed for synchronous optimization, and extending it to delayed communication would require nontrivial modifications because stale updates interact with its Nesterov acceleration and adaptive restart mechanisms. In addition, MESA is not included in the main experiment as it considers a different edge-based communication regime [7]. However, in Table III, we report a separate experiment that evaluates CORD against

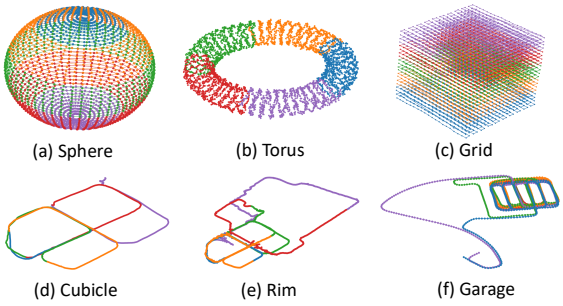

Fig. 2: Trajectory estimates returned by the proposed algorithm on benchmark datasets. Different colors correspond to different robots.

MESA under the edge-based regime as in [7].

A. Evaluation under Synchronous Communication

In this subsection, we perform experiments under the synchronous regime without communication delay. All methods are initialized using the same initialization scheme as implemented in AMM-PGO [8]. In Tab. I and Tab. II, we report the cost achieved by each method after 100 iterations. The costs of the initial and reference solutions are shown as $\mathcal{C}^{(0)}$ and $\mathcal{C}^*$, respectively. In addition, Tab. I also show the size of each dataset including the number of nodes and edges. CORD achieved the lowest error across nearly all 3D benchmark datasets for both metrics, with the exception of the `Sphere` dataset where AMM-PGO performed better, and the `Rim` dataset where MESA was better. Notably, our method outperforms AMM-PGO on the `Cubicle`, `Rim`, and `Garage`, demonstrating the validity of our algorithm on real-world data. Across all chordal datasets, CORD consistently achieves lower error than the DJ baseline, indicating a strict improvement over first-order preconditioned updates. This gap highlights the benefit of the proposed dynamical system formulation, where inertial and damping effects lead to faster convergence than purely gradient-based iterations.

Following [8], we also report the performance profile [42], which visualizes the percentage of problems solved by as a function of iterations. Given tolerance Δ, we introduce a threshold for determining if each problem is solved as $\mathcal{C}_\Delta = \mathcal{C}^* + \Delta(\mathcal{C}^{(0)} - \mathcal{C}^*)$. At each iteration, if the cost falls below the threshold, the dataset is regarded as solved by that method. The results for evaluated tolerances $\Delta = 0.005$ are presented in Fig. 3. We plot the performance profiles over 1000 iterations. The results indicate that CORD and AMM-PGO exhibit comparable performance, whereas DJ yields the smallest area under the curve (AUC), as expected. For chordal distance, CORD achieves an AUC of 851.36, comparable to AMM-PGO's 855.93. Notably, while AMM-PGO is specialized to chordal PGO, our framework is versatile to handle other cost functions including geodesic distance formulation. In the case of geodesic distance, CORD significantly outperforms the baseline algorithms, with the exception of `Rim` dataset. In particular, MESA is not included in the performance profile, as we observe that with its default parameter settings the method converges prematurely across all datasets and fails to reach the prescribed accuracy threshold.

TABLE II: Optimality gap achieved by synchronous methods after 100 iterations, under the geodesic distance formulation of PGO. Best and second best methods shown in **bold** and underline, respectively.

Dataset	$\mathcal{C}^{(0)}$	$\mathcal{C}^*$	$\mathcal{C}^{(100)}$		
			MESA [7]	DJ	CORD
Sm. Grid	6.5821×10^2	3.4130×10^2	3.5365×10^2	3.4414×10^2	$\mathbf{3.4320 \times 10^2}$
Sphere	7.6392×10^2	5.7657×10^2	5.8044×10^2	5.7719×10^2	$\mathbf{5.7671 \times 10^2}$
Torus	9.2693×10^3	8.9719×10^3	8.9770×10^3	8.9736×10^3	$\mathbf{8.9720 \times 10^3}$
Grid	3.3123×10^4	3.0968×10^4	3.0986×10^4	3.0973×10^4	$\mathbf{3.0968 \times 10^4}$
Cubicle	3.9095×10^2	3.2086×10^2	3.2337×10^2	3.2383×10^2	$\mathbf{3.2226 \times 10^2}$
Rim	3.8709×10^3	2.2367×10^3	$\mathbf{2.2476 \times 10^3}$	2.3991×10^3	2.2753×10^3
Garage	0.7083×10^0	0.6234×10^0	0.6582×10^0	0.6309×10^0	$\mathbf{0.6252 \times 10^0}$

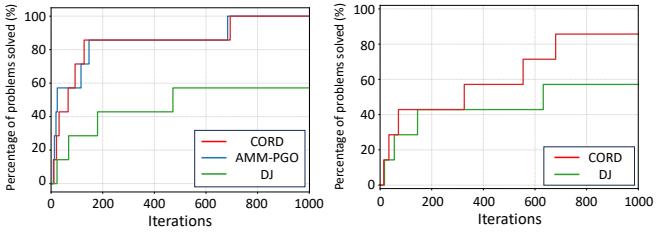

Fig. 3: Performance profiles of synchronous methods over 1,000 iterations with an evaluation tolerance of $\Delta = 0.005$ on 3D SLAM benchmark datasets. Left and right figures show chordal and geodesic PGO evaluations, respectively.

The dominant cost in solving distributed PGO is approximately solving a local pose graph using second-order information each iteration. In CORD and DJ this appears as the Hessian inversion (Algorithm 2, Line 12); in AMM-PGO [8] and MESA [7] it appears as minimizing the local majorized or augmented Lagrangian objective. To quantify the practical overhead, we report average iteration runtime and communication packet size on the largest `Rim` dataset using 32-bit floats, and compare against DJ since other baselines are implemented in C++. CORD requires 101.9 ms (vs. 43.0 ms for DJ) and 5.9 KB (vs. 3.2 KB for DJ), while using 80 iterations (vs. 924 for DJ) to achieve $\Delta = 10^{-2}$. This shows a favorable trade-off since CORD achieves substantially faster convergence while incurring modest overhead.

B. Evaluation on Asynchronous Communication

We validated our approach under asynchronous conditions using both public benchmarks and simulation data. For the simulation, we evaluate on a different and more challenging initialization condition. Specifically, we obtain the initial solution by propagating odometry measurements for each robot starting from its known initial pose in the global frame.

For the public benchmarks, a uniform delay of 5 steps is applied across all datasets, and results are summarized in the "Async." column of Tab. I. In the presence of communication delays, the use of outdated neighbor poses induces larger errors in the inter-robot edge gradients compared to the synchronized case. This error acts as a perturbation to the denominator in (12). Consequently, maintaining the same Δt as in the

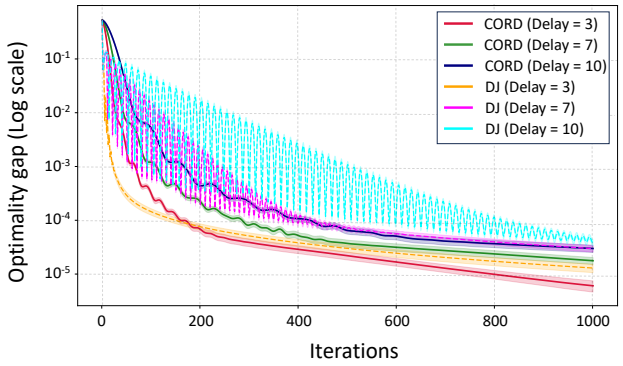

Fig. 4: Optimality gap for varying delay steps in asynchronous simulations, plotted on a log scale. Solid lines indicate the mean performance averaged over 20 Monte Carlo runs, and shaded areas represent the standard deviation.

TABLE III: Evaluation under edge-based regime and geodesic distance. CORD uses a constant set of parameters: $d = 4$, $m = 0.7$, $\Delta t = 0.1$.

Method	Sphere	Torus	Garage
MESA [7]	7.1935×10^2	1.3321×10^4	0.7045×10^0
CORD (const.)	6.9539×10^2	1.2158×10^4	0.6537×10^0

synchronized setting would compromise convergence guarantees. To mitigate this, we employ a conservative step size by reducing Δt by more than 90%. Despite this restriction, CORD consistently achieved a lower final cost than DJ. The inherent inertia allows the system to achieve acceleration even under delayed communication, thereby compensating for the reduced step size and facilitating faster convergence.

To provide further insights on the algorithm performance, we include additional simulation experiments with varying delay conditions. We simulate a multi-robot SLAM scenario shown in Fig. 1 with four robots where each robot follows a 3D grid trajectory consisting of 125 nodes. The robots are distributed in a two by two grid formation. Loop closures were generated within a 1.4 m radius with a probability of 0.2 for intra-robot edges and 0.3 for inter-robot edges. Measurements were corrupted by Gaussian noise where standard deviations were uniformly sampled from ranges of 0.05 m to 0.15 m for translation and 1.0 degree to 3.0 degrees for rotation for intra-robot edges. For inter-robot edges, we employ higher noise ranges of 0.10 m to 0.30 m for translation and 3.0 degrees to 10.0 degrees for rotation. Fig. 4 shows the optimality gap over 1000 iterations. Regarding DJ, using the standard step size of 1.0 causes severe oscillations and divergence. Similar to ASAPP [10], we empirically determine the maximum step size for DJ that ensured convergence through manual tuning. Overall, CORD achieved a lower final cost than DJ across all delay steps. When comparing performance under identical delay conditions, CORD initially exhibits a higher cost due to the lower step size. However, the DJ trajectory exhibits oscillations proportional to the delay length, leading to significant slowdown in convergence. This vulnerability stems from the fully parallel nature of the DJ update scheme, which is highly susceptible to outdated neighbor information.

TABLE IV: Ablation study results reporting optimality gap. `State Dep.`: using state-dependent mass M and damping D; `Nbr. Pred`: using predicted neighbor pose from velocity-based extrapolation. Results are shown as Mean $\pm$ Std. Dev. across 3 datasets.

State Dep.	Nbr. Pred.	Opt. Gap $\downarrow$ (Mean $\pm$ Std.)
$\times$	$\times$	$1.53 \times 10^{-2} \pm 0.23 \times 10^{-2}$
✓	$\times$	$1.61 \times 10^{-2} \pm 0.18 \times 10^{-2}$
$\times$	✓	$0.29 \times 10^{-2} \pm 0.08 \times 10^{-2}$
✓	✓	$\mathbf{0.28 \times 10^{-2}} \pm 0.09 \times 10^{-2}$

As communication delays increase, the discrepancy between the available neighbor poses and their actual states increases. This mismatch induces erroneously large gradients along inter-robot edges, leading to the instability of system. In contrast, even though the proposed CORD framework also operates in a parallel manner, it remains robust to such perturbations due to the built-in inertia and damping of the system. Furthermore, by utilizing the communicated body velocities to predict the current states of neighbors, our method effectively compensates for communication latencies. This predictive mechanism results in the smooth convergence trajectory observed in the results, distinct from the oscillatory behavior of DJ.

We further evaluate CORD under more challenging asynchronous communication conditions beyond constant delays. Specifically, we introduce 10% random packet loss and randomized communication delays in the range $[1, 10]$ while using the same fully parallel asynchronous update protocol and a single constant parameter setting, $d = 4$, $m = 0.7$, and $\Delta t = 0.2$. As shown in the last column of Table I, CORD remains stable and continues to reduce the objective under these non-ideal communication conditions. These results provide additional evidence that the proposed dynamics are robust to packet loss and heterogeneous delays, beyond the constant-delay settings considered above.

C. Evaluation under Edge-Based Communication

To further evaluate CORD under an alternative distributed communication protocol, we adapt CORD to the edge-based regime used by MESA and compare on the three datasets considered in [7]. In this regime, one pair of neighboring robots communicates at each iteration without delay, providing a complementary setting to the fully parallel asynchronous protocol considered above. As shown in Table III, CORD achieves lower costs after 100 iterations using a single constant parameter setting, $d = 4$, $m = 0.7$, and $\Delta t = 0.1$.

D. Ablation Study

In this ablation study, we investigate the impact of different features of the proposed algorithm. Specifically, we compare our approach against two variants: (i) fixing the mass and damping term M, D based on the initial Hessian, and (ii) using received neighbor poses directly without the predictive extrapolation as in line 6 of Algorithm 2. For this purpose, we generate a set of datasets using the same simulation suite described in Sec. VI-B and set a fixed communication delay of 7 steps. Performance was evaluated using the chordal distance optimality gap over 1000 iterations. As shown in Tab. IV,

while the state-dependent M and D yield generally better performance, the improvement is overall marginal. This validates the constant setting as a competitive alternative that provides additional computational efficiency. In contrast, our results show that the neighbor pose prediction plays a critical role. Omitting the velocity-based prediction causes the system to diverge due to inaccurate neighbor information. This instability arises because large residuals in inter-robot edges increase the gradient ∇C_k in the denominator of (12). Consequently, a small value of Δt is needed to prevent divergence, which inevitably leads to a higher final cost.

VII. Discussion

A. Constant Mass Assumption

While the proposed CORD framework demonstrates strong performance in both synchronous and asynchronous distributed PGO settings, the current convergence analysis is restricted to the synchronous case with a constant mass matrix. Extending the theory to explicitly account for communication delays and state-dependent $M(X)$ would further close the gap between the analysis and the practical implementation.

Nevertheless, we empirically observe that the constant-mass analysis remains informative for the state-dependent variant. Near the attained minimizers, the mass matrix changes slowly, and the resulting dynamics can be viewed as a small perturbation of the analyzed system. To support this observation, Fig. 5 reports the relative mass variation, $\|M^k - M^{k-1}\|_F / \|M^k\|_F$, together with the normalized energy convergence, $(E^k - E^*)/E^*$, for the state-dependent setup with a constant set of parameters. After a short initial transient, the relative mass variation remains below 0.1% for most iterations across all datasets, and the energy decreases monotonically except for a minor initial overshoot.

A natural next step would be to formalize both state dependence and communication delay as bounded perturbations of the current dynamics, thereby extending the convergence guarantees to more realistic distributed settings.

B. Parameter Tuning

In the current implementation, the CORD parameters are selected according to their dynamical roles: the damping coefficient d controls early-stage energy dissipation, the mass m affects acceleration, and Δt determines the maximum integration step size while ensuring numerical stability. Although some experiments use dataset-specific parameter tuning, we observe that CORD is not highly sensitive to the choice of parameters. In particular, a single constant setting, $d = 2$, $m = 0.8$, and $\Delta t = 1$, achieves performance close to the per-dataset tuned setting in the synchronous chordal case, as shown in Table I. Under delayed communication, we further observe that m and d remain largely transferable across datasets, while Δt should be reduced as the communication delay increases; see Table VII in the appendix. This behavior is consistent with the theoretical insight from ASAPP [10], where smaller step sizes improve stability under asynchronous updates. Adaptive parameter selection is an important direction for future work.

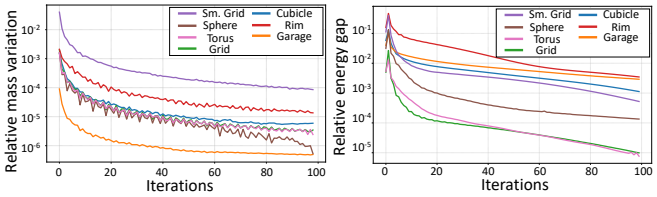

Fig. 5: Relative mass variation and energy convergence (state-dependent, synchronous, chordal) with constant parameters $d = 2, m = 0.8, \Delta t = 1$.

C. Extensions to General Factor Graphs

Although this work is formulated in the context of PGO, the underlying continuous-time Riemannian dynamics are not restricted to pairwise pose measurements. Because our formulation builds upon the continuous-time dynamics on matrix Lie groups studied in [11], it inherently applies to spaces beyond SE(3), although it does not generalize to arbitrary Riemannian manifolds. This theoretical foundation opens up an exciting direction for future work on distributed factor graph optimization [38] that involves optimization variables over other matrix Lie groups.

VIII. Conclusion

We presented CORD, a novel framework for distributed PGO by formulating the problem as a second-order continuous dynamical system on Riemannian manifolds. By modeling the optimization variables as massive particles subject to dissipation, we showed that the equilibrium states of the system correspond to the first-order critical points of the original optimization problem. Our derived equations of motion generalize classical update rules like Gauss-Newton and Riemannian Gradient Descent. Furthermore, we presented a convergence analysis of the proposed algorithm, and established conditions that guarantee energy dissipation after discretization. Numerical evaluations on public benchmarks and simulation datasets validated the effectiveness of our approach. In synchronous settings, the proposed method achieved accuracy comparable to state-of-the-art distributed algorithms. More importantly, in asynchronous scenarios with varying communication delays, our solver demonstrated superior resilience and faster convergence speed, confirming the practical benefits of the proposed dynamical system-based solver.

Acknowledgments

M. Ghaffari was supported by AFOSR MURI FA9550-23-1-0400 and AFOSR YIP FA9550-25-1-0224.

References

[1] Kamak Ebadi, Lukas Bernreiter, Harel Biggie, Gavin Catt, Yun Chang, Arghya Chatterjee, Christopher E. Denniston, Simon-Pierre Deschênes, Kyle Harlow, Shehryar Khattak, Lucas Nogueira, Matteo Palieri, Pavel Petráček, Matěj Petrlík, Andrzej Reinke, Vít Krátký, Shibo Zhao, Ali-akbar Agha-mohammadi, Kostas Alexis, Christoffer Heckman, Kasra Khosoussi, Navinda Kottege, Benjamin Morrell, Marco Hutter, Fred Pauling, François Pomerleau, Martin Saska, Sebastian Scherer, Roland Siegwart, Jason L. Williams, and Luca Carlone. Present and Future of SLAM in Extreme Environments: The DARPA SubT Challenge. *IEEE Trans. on Robotics*, 40:936–959, 2024. ISSN 1941-0468. doi: 10.1109/TRO.2023.3323938.

[2] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P How, and Luca Carlone. Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems. *IEEE Transactions on Robotics*, 38(4), 2022.

[3] Xu Liu, Jiuzhou Lei, Ankit Prabhu, Yuezhan Tao, Igor Spasojevic, Pratik Chaudhari, Nikolay Atanasov, and Vijay Kumar. SlideSLAM: Sparse, Lightweight, Decentralized Metric-Semantic SLAM for Multi-Robot Navigation, July 2024. arXiv:2406.17249.

[4] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-SLAM : Sparse Decentralized Collaborative Simultaneous Localization and Mapping Framework for Multi-Robot Systems. *IEEE Robotics and Automation Letters*, 9(1):475–482, January 2024. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2023.3333742.

[5] Patrik Schmuck, Thomas Ziegler, Marco Karrer, Jonathan Perraudin, and Margarita Chli. COVINS: Visual-Inertial SLAM for Centralized Collaboration. In *2021 IEEE Int. Symposium on Mixed and Augmented Reality Adjunct*, pages 171–176, October 2021. doi: 10.1109/ISMAR-Adjunct54149.2021.00043.

[6] Yulun Tian, Kasra Khosoussi, David M. Rosen, and Jonathan P. How. Distributed Certifiably Correct Pose-Graph Optimization. *IEEE Trans. on Robotics*, 37(6):2137–2156, December 2021. ISSN 1941-0468. doi: 10.1109/TRO.2021.3072346.

[7] Daniel McGann, Kyle Lassak, and Michael Kaess. Asynchronous Distributed Smoothing and Mapping via On-Manifold Consensus ADMM. In *IEEE Int. Conf. on Robotics and Automation*, pages 4577–4583, 2024.

[8] Taosha Fan and Todd D. Murphey. Majorization Minimization Methods for Distributed Pose Graph Optimization. *IEEE Trans. on Robotics*, 40:22–42, 2024. ISSN 1941-0468. doi: 10.1109/TRO.2023.3324818.

[9] Riku Murai, Joseph Ortiz, Sajad Saeedi, Paul H. J. Kelly, and Andrew J. Davison. A Robot Web for Distributed Many-Device Localization. *IEEE Trans. on Robotics*, 40:121–138, 2024. ISSN 1941-0468. doi: 10.1109/TRO.2023.3324127.

[10] Yulun Tian, Alec Koppel, Amrit Singh Bedi, and Jonathan P. How. Asynchronous and Parallel Distributed Pose Graph Optimization. *IEEE Robotics and Automation Letters*, 5(4):5819–5826, October 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.3010216.

[11] Anthony Bloch, PS Krishnaprasad, Jerrold E Marsden, and Tudor S Ratiu. The euler-poincaré equations and double bracket dissipation. *Communications in mathematical physics*, 175(1): 1–42, 1996.

[12] Thai Duong, Abdullah Altawaitan, Jason Stanley, and Nikolay Atanasov. Port-hamiltonian neural ode networks on lie groups for robot dynamics learning and control. *IEEE Transactions on Robotics*, 2024.

[13] Sangli Teng, Ashkan Jasour, Ram Vasudevan, and Maani Ghaffari. Convex geometric motion planning of multi-body systems on lie groups via variational integrators and sparse moment relaxation. *The International Journal of Robotics Research*, 44 (10-11):1784–1813, 2025.

[14] Alexander Cunningham, Manohar Paluri, and Frank Dellaert. DDF-SAM: Fully distributed SLAM using Constrained Factor Graphs. In *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3025–3030, 2010. doi: 10.1109/IROS.2010.5652875.

[15] Alexander Cunningham, Vadim Indelman, and Frank Dellaert. DDF-SAM 2.0: Consistent distributed smoothing and mapping. In *2013 IEEE Int. Conf. on Robotics and Automation*, pages 5220–5227, 2013. doi: 10.1109/ICRA.2013.6631323.

[16] Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Henrik I Christensen, and Frank Dellaert. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *The Int. Journal of Robotics Research*, 36(12):1286–1311, October 2017. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364917732640.

[17] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. *The Int. Journal of Robotics Research*, 38(2-3):95–125, 2019. doi: 10.1177/0278364918784361.

[18] Jesus Briales and Javier Gonzalez-Jimenez. Cartan-sync: Fast and global se (d)-synchronization. *IEEE Robotics and Automation Letters*, 2(4):2127–2134, 2017.

[19] Cunhao Li, Guanghui Guo, Peng Yi, and Yiguang Hong. Distributed Pose-Graph Optimization With Multi-Level Partitioning for Multi-Robot SLAM. *IEEE Robotics and Automation Letters*, 9(6):4926–4933, June 2024. ISSN 2377-3766. doi: 10.1109/LRA.2024.3382531.

[20] D. McGann and M. Kaess. iMESA: Incremental distributed optimization for collaborative simultaneous localization and mapping. In *Proc. Robotics: Science and Systems (RSS)*, Delft, NL, 2024.

[21] Philipp Bänninger, Ignacio Alzugaray, Marco Karrer, and Margarita Chli. Cross-agent relocalization for decentralized collaborative slam. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5551–5557. IEEE, 2023.

[22] Xiangyu Liu and Margarita Chli. Distributed pose graph optimization via contractive belief sharing. *IEEE Robotics and Automation Letters*, 2026.

[23] Weijie Su, Stephen Boyd, and Emmanuel J Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. *Journal of Machine Learning Research*, 2016.

[24] Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, 1983.

[25] Andre Wibisono, Ashia C Wilson, and Michael I Jordan. A variational perspective on accelerated methods in optimization. *proceedings of the National Academy of Sciences*, 113(47), 2016.

[26] Ashia C Wilson, Ben Recht, and Michael I Jordan. A lyapunov analysis of accelerated methods in optimization. *Journal of Machine Learning Research*, pages 1–34, 2021.

[27] Michael Betancourt, Michael I Jordan, and Ashia C Wilson. On symplectic optimization. *arXiv preprint arXiv:1802.03653*, 2018.

[28] Valentin Duruisseaux and Melvin Leok. Practical perspectives on symplectic accelerated optimization. *Optimization Methods and Software*, 38(6):1230–1268, 2023.

[29] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Structure-preserving algorithms for ordinary differential equations. *Geometric numerical integration*, 31, 2006.

[30] Adrian Hauswirth, Saverio Bolognani, Gabriela Hug, and Florian Dörfler. Projected gradient descent on riemannian manifolds with applications to online power system optimization. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016.

[31] Valentin Duruisseaux and Melvin Leok. Accelerated optimization on riemannian manifolds via discrete constrained variational

integrators. *Journal of Nonlinear Science*, 32(4):42, 2022.

[32] Vladislav Golyanik, Sk Aziz Ali, and Didier Stricker. Gravitational approach for point set registration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[33] Philipp Jauer, Ivo Kuhlemann, Ralf Bruder, Achim Schweikard, and Floris Ernst. Efficient registration of high-resolution feature enhanced point clouds. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1102–1115, 2018.

[34] Vladislav Golyanik, Christian Theobalt, and Didier Stricker. Accelerated gravitational point set alignment with altered physical laws. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2080–2089, 2019.

[35] Sk Aziz Ali, Vladislav Golyanik, and Didier Stricker. Nrga: Gravitational approach for non-rigid point set registration. In *2018 International Conference on 3D Vision (3DV)*, pages 756–765. IEEE, 2018.

[36] Mingyang Zhao, Lei Ma, Xiaohong Jia, Dong-Ming Yan, and Tiejun Huang. Graphreg: Dynamical point cloud registration with geometry-aware graph signal processing. *IEEE Transactions on Image Processing*, 31:7449–7464, 2022.

[37] Heng Yang, Chris Doran, and Jean-Jacques Slotine. Dynamical pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5926–5935, 2021.

[38] Frank Dellaert and GTSAM Contributors. borglab/gtsam, May 2022. URL https://github.com/borglab/gtsam.

[39] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[40] Nicolas Boumal, Pierre-Antoine Absil, and Coralia Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2019.

[41] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International conference on machine learning*, pages 3043–3052. PMLR, 2018.

[42] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.