Hyperion – A fast, versatile symbolic Gaussian Belief Propagation framework for Continuous-Time SLAM

David Hug[1] , Ignacio Alzugaray[2] , and Margarita Chli[1,3]

[1] Vision for Robotics Lab, ETH Zürich, Zürich, Switzerland
[2] Dyson Robotics Lab, Imperial College London, London, United Kingdom
[3] Vision for Robotics Lab, University of Cyprus, Nicosia, Cyprus

Abstract. Continuous-Time Simultaneous Localization And Mapping (CTSLAM) has become a promising approach for fusing asynchronous and multi-modal sensor suites. Unlike discrete-time SLAM, which estimates poses discretely, CTSLAM uses continuous-time motion parametrizations, facilitating the integration of a variety of sensors such as rolling-shutter cameras, event cameras and Inertial Measurement Units (IMUs). However, CTSLAM approaches remain computationally demanding and are conventionally posed as centralized Non-Linear Least Squares (NLLS) optimizations. Targeting these limitations, we not only present the fastest SymForce-based [22] B- and Z-Spline implementations achieving speedups between 2.43x and 110.31x over Sommer *et al.* [40] but also implement a novel continuous-time Gaussian Belief Propagation (GBP) framework, coined Hyperion, which targets decentralized probabilistic inference across agents. We demonstrate the efficacy of our method in motion tracking and localization settings, complemented by empirical ablation studies.
Code: https://github.com/VIS4ROB-lab/hyperion

Keywords: Gaussian Belief Propagation · Continuous-Time SLAM · Distributed Non-Linear Least Squares Optimization · B- and Z-Splines

1 Introduction

Estimating a sensor-suite's ego-motion and workspace employing Simultaneous Localization And Mapping (SLAM) techniques, has long been studied using a wide variety of sensing modalities, such as vision sensors [35,39,43,46], IMUs [7,18,36], Global Positioning System (GPS) feeds [21,23], and laser ranging [9] sensors. Unlike traditional, discrete-time approaches, which require careful synchronization of sensory measurements due to the discretization of motion-parametrizing states, Continuous-Time Simultaneous Localization And Mapping (CTSLAM) [9,12,13, 15,33] offers native support for the fusion of asynchronous measurements due to its continuous-time parametrization that yields pose, velocity and acceleration estimates at arbitrary instances in time.

Despite their advantages, CTSLAM approaches often entail higher computational complexity than conventional approaches, which somewhat hinders their
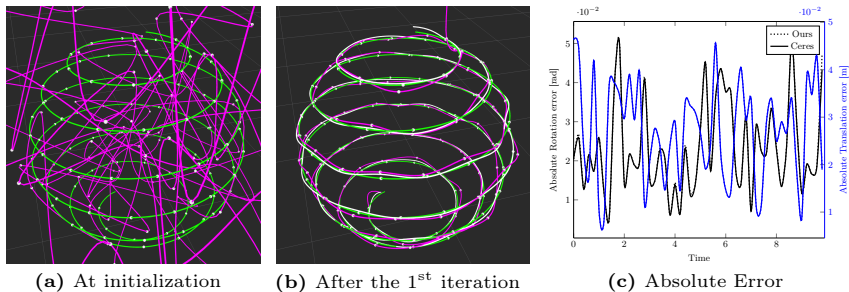
(a) At initialization   (b) After the 1st iteration   (c) Absolute Error

Fig. 1: Both the proposed continuous-time GBP solver (in magenta) and the conventional NLLS solver [1] (in white) converge to identical solutions close to the ground truth (in green) even under poor initialization (± 1.00 m/rad) and substantial pose measurement noise (± 0.05 m/rad).

deployment in real-world scenarios. In addition, most (discrete- and continuous-time) SLAM systems pose the underlying optimization as a centralized Non-Linear Least Squares (NLLS) problem, which, without further modifications, strictly limits their applicability to single-agent setups.

Thus, novel, decentralized algorithms along with more effective continuous-time motion parametrizations are paramount in the quest to overcome existing challenges in traditional, centralized CTSLAM techniques. Distributed methods such as GBP are especially promising given that they, in contrast to conventional NLLS paradigms, achieve iterative, probabilistic inference through message-passing between individual nodes and factors in a factor graph and that they operate in a distributed and asynchronous manner by nature rendering them inherently scalable, even across multiple agents. In addition, GBP also explicitly models the uncertainties of optimizable quantities, which can be leveraged to selectively direct computational resources to non-converged nodes in the graph. This circumstance is especially promising in the context of CTSLAM where effective allocation of computational resources towards the least certain estimates promises to further reduce their computational complexity.

This work, in particular, pushes the performance envelope of B- and Z-Splines and devises an innovative, distributed optimization strategy for CTSLAM using GBP. Whilst the need for faster continuous-time parametrizations is self-evident, we also specifically target the absence of a distributed, continuous-time state estimation framework. Such a framework, not only promises flexible resource allocation and decentralized state estimation but also allows fusing asynchronous measurements. Here, we showcase the practicality of the proposed continuous-time GBP framework and provide it as an open-source implementation to encourage benchmarking. In addition to these contributions, this work also

- offers the fastest, fully analytic B- and Z-Spline implementations to date,
- presents a novel, symbolic GBP-based framework for continuous-time SLAM,
- demonstrates the suitability of the proposed method in practical setups, and
- provides detailed ablation studies on the algorithm itself.

2   Related Work

The SLAM problem has long been researched as it comprises the core of robotic perception, with seminal works focusing on discrete-time, monocular [26, 36] and stereo setups [35, 37], as well as works which propose fusion of additional sensing information, such as inertial [5–7, 19], and laser ranging [9]. In addition to these feature-based approaches to SLAM, alternative powerful paradigms have been proposed, such as direct methods [10, 11] and machine-learning based techniques [24, 44] have been demonstrated to be beneficial in some scenarios.

In contrast to traditional, discrete-time SLAM methods that have been widely adopted, CTSLAM techniques have early on been demonstrated to exhibit great potential for high-fidelity and continuous estimates of motion [2, 12, 13, 15, 20, 25, 42, 45], albeit posing fundamental scientific and algorithmic challenges. The key advantage of these methods lies in their inherent capability to fuse unsynchronized and asynchronous measurements (*e.g.* from a rolling shutter camera or an event-based vision sensor) in the estimation processes, in contrast to conventional approaches. However, the wider adoption of CTSLAM techniques remains impeded by challenges in finding more efficient motion representations, addressing convergence issues as well as overcoming computational limitations, missing out on the promise for high-fidelity motion and scene estimation.

Following promising leads in fundamental SLAM research for single agents, a novel challenge and desire to deploy the same techniques in collaborative, multi-agent setups arose. Works such as [38] proposed means to address centralized multi-agent SLAM, while [18] investigated the advantages of variable-stereo baseline setups, which leverage the views from two agents to boost the accuracy of high-altitude depth estimates. A common approach to solve NLLS optimizations in a distributed fashion is based on Alternating Direction Method of Multipliers (ADMM) approaches similar to the work in [34] and [3], which leverage the use of dual residuals to ensure consistent estimates across distributed multi-agent NLLS optimizations. Orthogonally, GBP approaches [8, 28, 30, 32] can perform distributed and asynchronous inference of states via an equivalent message-passing scheme, making them ideal candidates to address multi-agent SLAM.

In this work, we build upon these ideas, consolidate a distributed GBP optimization with a continuous-time parametrization, and demonstrate the suitability of the novel, combined method in practical setups.

3   Methodology

3.1   Preliminaries

All Simultaneous Localization And Mapping (SLAM) algorithms, in essence, aim to estimate an optimal, cost-minimizing set of optimizable parameters $\boldsymbol{\Theta}$ which are set to describe a collection of noisy sensory measurements $\boldsymbol{m}$ with the highest possible accuracy. Specifically, in its NLLS formulation, this optimal solution is found by minimization of the cumulative cost over a set of weighted residuals $\bar{\boldsymbol{r}}$ associated with measurements $\boldsymbol{m}$ stemming from a connected sensor
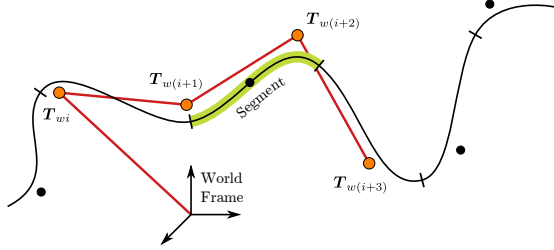
Fig. 2: For every (valid) instance in time t, a collection of adjacent bases (in orange) gives rise to an individual segment (in green) of a cubic B-Spline. An interpolated pose at query time t is then obtained from the (cumulative) blending of these bases.

s. A weighted residuals $\bar{r}$ at measurement time t is computed by comparing a predicted measurement $\hat{m}(t, \theta_s)$ with sensor-associated parameters $\theta_s \subseteq \Theta$ to a measured one $m(t)$ through the application of a metric μ (*i.e.* $\hat{x} \boxminus_\mu x$) and a subsequent weighting *via* the square-root information Ω_m according to

$$r(t, \theta_s) = \hat{m}(t, \theta_s) \boxminus_\mu m(t) \text{ and} \tag{1}$$

$$\|\bar{r}\|^2 = \bar{r}^\top \bar{r} = r^\top \Omega_m^\top \Omega_m r = r^\top \Lambda_m r = r^\top \Sigma_m^{-1} r, \tag{2}$$

where Σ_m and Λ_m are the covariance and precision matrix, respectively. Ultimately, we aim to minimize the sum over residuals stemming from all connected sensors $\mathcal{S}$ and their associated measurements $\mathcal{M}_s$ with measurement times $\mathcal{T}_s$. To this end, established NLLS solvers [1] are commonly deployed in practice, which aim to obtain the optimal, cost-minimizing parameters Θ^* as defined below.

$$\Theta^* = \underset{\Theta}{\mathrm{argmin}} \left[\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \frac{1}{2} \|\bar{r}(t, \theta_s)\|^2 \right]. \tag{3}$$

3.2 Continuous-Time Motion

In recent times, continuous-time formulations of conventional SLAM algorithms have been studied by multiple authors [12, 13, 15, 16, 20, 40], where the most common motion parametrization is based on the concept of cubic B-Splines. This practice is rooted in several favorable properties of B-Splines, such as their compact representation, finite support, analytic Jacobians, and their $\mathcal{C}_2$-continuity for predicting instantaneous velocities and accelerations. In this work, we utilize split interpolations for world-to-body transformations $T_{wb}(t) \in \mathbb{SE}(3)$, illustrated in Fig. 2, that separately parametrize rotations $R_{wb}(t) \in \mathbb{SO}(3)$ and translations $t_{wb}(t) \in \mathbb{R}^3$. In particular, we compute transformations $T_{wb}(t)$ using

$$T_{wb}(t) = \begin{bmatrix} R_{wb}(q_{wb}(t)) & t_{wb}(t) \\ 0 & 1 \end{bmatrix} \in \mathbb{SE}(3) \text{ with} \tag{4}$$

$$\boldsymbol{q}_{wb}(t) = \boldsymbol{q}_{wi} * \prod_{j=1}^{k} \left(\boldsymbol{q}_{w(i+j-1)}^{-1} * \boldsymbol{q}_{w(i+j)} \right)^{\lambda_j(t)} \tag{5}$$

$$\boldsymbol{t}_{wb}(t) = \boldsymbol{t}_{wi} + \sum_{j=1}^{k} \left[\lambda_j(t) \left(\boldsymbol{t}_{w(i+j)} - \boldsymbol{t}_{w(i+j-1)} \right) \right], \tag{6}$$

where individual evaluations of $\boldsymbol{T}_{wb}(t)$ depend on a collection of bases $\{\mathcal{B}_i, \ldots, \mathcal{B}_{i+k}\}$ (see Fig. 2) and each basis $\mathcal{B}_i$ comprises a time t_i, a quaternion $\boldsymbol{q}_{wi}$ and a translation $\boldsymbol{t}_{wi}$. Above, k is the Degree of Freedom (DoF) of the B-/Z-Spline, rotations $\boldsymbol{R}_{wb}(t)$ are parametrized by quaternions $\boldsymbol{q}_{wb}(t)$ and the expression $\lambda_j(t)$ serves as a placeholder for concrete interpolation coefficients found in [4, 15, 17, 20].

3.3 Continuous-Time Optimization

In its probabilistic formulation [30, 32], the non-linear minimization problem from Eq. (3) is equivalent to finding an optimal probability distribution that accounts for all acquired measurements. Without loss of generality, one can also represent the problem as a product of factors $f_i \propto e^{-E_i(\boldsymbol{\theta}_i)}$ with induced energies $E_i(\boldsymbol{\theta}_i)$, arriving at the modified expression to obtain the optimal parameters $\boldsymbol{\Theta}^*$.

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\mathrm{argmax}} \log\left(p\left(\boldsymbol{\Theta}\right)\right) = \underset{\boldsymbol{\Theta}}{\mathrm{argmin}} \sum_i E_i(t_i, \boldsymbol{\theta}_i) \quad \text{with} \tag{7}$$

$$p\left(\boldsymbol{\Theta}\right) = \prod_i f_i\left(t_i, \boldsymbol{\theta}_i\right) \propto \prod_i e^{-E_i(t_i, \boldsymbol{\theta}_i)}. \tag{8}$$

In this work, we use multi-variant Gaussians $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ to model the factors f_i in Eq. (8), linking the energies in Eq. (7) to the residuals in Eq. (2) according to

$$E_i(t_i, \boldsymbol{\theta}_i) = \|\bar{\boldsymbol{r}}_i\|^2 = \boldsymbol{r}_i^\top(t_i, \boldsymbol{\theta}_i) \boldsymbol{\Omega}_i^\top \boldsymbol{\Omega}_i \boldsymbol{r}_i(t_i, \boldsymbol{\theta}_i). \tag{9}$$

The generic residuals $\bar{\boldsymbol{r}}_i$ themselves are non-linear with a corresponding Taylor expansions in $\boldsymbol{\theta}_i$ around some linearization point $\boldsymbol{\theta}_i^0$ such that

$$\bar{\boldsymbol{r}}_i(\boldsymbol{\theta}_i) - \bar{\boldsymbol{r}}_i(\boldsymbol{\theta}_i^0) \approx D\bar{\boldsymbol{r}}_i(\boldsymbol{\theta}_i^0)\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^0\right) = \bar{\boldsymbol{J}}_i^0\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^0\right) = \bar{\boldsymbol{J}}_i^0 \boldsymbol{\tau}_i^0. \tag{10}$$

Furthermore, all factors f_i, their energies $E_i(t_i, \boldsymbol{\theta}_i)$ respectively, can also be converted to an equivalent, incremental information form $\mathcal{N}^{-1}(\boldsymbol{\eta}_i^0, \boldsymbol{\Lambda}_i^0)$, yielding

$$E_i(t_i, \boldsymbol{\tau}_i^0) \approx \frac{1}{2}\boldsymbol{\tau}_i^{0,\top}\boldsymbol{\Lambda}_i^0\boldsymbol{\tau}_i^0 - \boldsymbol{\tau}_i^{0,\top}\boldsymbol{\eta}_i^0 \quad \text{where} \tag{11}$$

$$\boldsymbol{\eta}_i^0 = -\bar{\boldsymbol{J}}_i^{0,\top}\bar{\boldsymbol{r}}_i^0 \quad \text{and} \quad \boldsymbol{\Lambda}_i^0 = \bar{\boldsymbol{J}}_i^{0,\top}\bar{\boldsymbol{J}}_i^0. \tag{12}$$

This dual representation of factors f_i and the existence of the Gaussian $\mathcal{N}$ and its inverse $\mathcal{N}^{-1}$ is vital for efficient conditioning and marginalization of individual terms in the GBP algorithm presented in the next section.

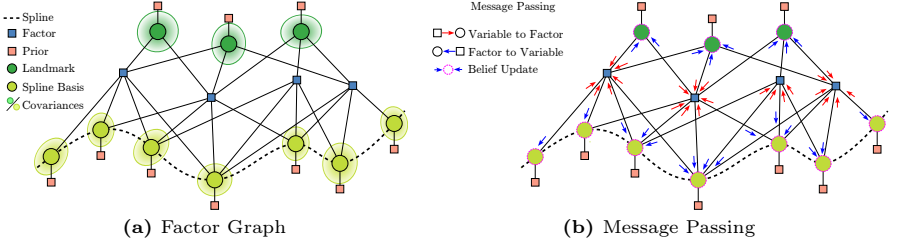**(a)** Factor Graph   **(b)** Message Passing

Fig. 3: (a) Qualitative illustration of the factor graphs resulting from a continuous-time motion parametrization. Notably, even one of the simplest factor archetypes, which purely relies on the motion-parameterizing nodes as well as some landmarks, introduces a considerable amount of loops in the continuous-time realm. This can be attributed to the fact that poses $\boldsymbol{T}_{wb}(t)$ depend on multiple bases for any given time t. (b) Visualization of the message passing algorithm between nodes and factors in a graph $\mathcal{G}$.

Gaussian Belief Propagation (GBP) As illustrated in Fig. 3a, Eqs. (3) and (7) expose equivalent visual representations as factor graphs, giving rise to compact and intuitive descriptions of complex probabilistic dependencies which are encoded into bipartite graphs $\mathcal{G}$ containing factors $f_i \sim \mathcal{N}^{-1}(\boldsymbol{\eta}_{f_i}, \boldsymbol{\Lambda}_{f_i})$ and nodes $n_j \sim \mathcal{N}(\boldsymbol{\mu}_{n_j}, \boldsymbol{\Sigma}_{n_j}) = \mathcal{N}^{-1}(\boldsymbol{\eta}_{n_j}, \boldsymbol{\Lambda}_{n_j})$. Intuitively, the GBP algorithm, thus, comprises two steps to iteratively solve for the optimal solution to Eq. (7), namely node and factor updates with intermediate factor-to-node $\boldsymbol{m}_{f_i \rightarrow n_j}$ and node-to-factor $\boldsymbol{m}_{n_j \rightarrow f_i}$ message passing detailed in the following.

Node Updates: The Gaussian nodes $n_j \in \mathcal{G}$ with neighborhood $N(n_j)$ of connected factors $f_i \in N(n_j)$ are updated by taking the product over incoming factor-to-node messages $\boldsymbol{m}_{f_i \rightarrow n_j}$, which reduces to a simple summation of Gaussians for (linear) vector spaces, resulting in node beliefs $B(n_j) = \mathcal{N}^{-1}(\boldsymbol{\eta}_{n_j}, \boldsymbol{\Lambda}_{n_j})$, where

$$\boldsymbol{\eta}_{n_j} = \boldsymbol{\eta}_{n_j}^p + \sum_{f_i \in N(n_j)} \boldsymbol{\eta}_{f_i \rightarrow n_j} \quad \text{and} \quad \boldsymbol{\Lambda}_{n_j} = \boldsymbol{\Lambda}_{n_j}^p + \sum_{f_i \in N(n_j)} \boldsymbol{\Lambda}_{f_i \rightarrow n_j}. \quad (13)$$

Above, $P(n_j) = \mathcal{N}^{-1}(\boldsymbol{\eta}_{n_j}^p, \boldsymbol{\Lambda}_{n_j}^p)$ denotes a known prior on the node belief $B(n_j)$ which can be leveraged to better constrain the problem in practice. However, despite its simplicity and elegance, this straightforward approach can not be applied to Lie groups G with associated tangent space $\mathfrak{g}$ and their operators $\boxminus : G \times G \mapsto \mathfrak{g}$ and $\boxplus : G \times \mathfrak{g} \mapsto G$ extensively used in robotics. Hence, a more sophisticated approach is required to obtain the product of factor-to-node messages and to handle frame conversions between elements in the Lie group.

To this end, we follow a similar strategy as Murai *et al.* [28] and propose the use of a Mixed Gaussian Representation (MGR) to parameterize Lie-group-valued nodes as $n_j \sim \mathfrak{N}(\boldsymbol{\mu}_{n_j}, \boldsymbol{\Lambda}_{n_j})$, where $\boldsymbol{\mu}_{n_j} \in G$ and $\boldsymbol{\Lambda}_{n_j} \in \mathbb{R}^{\dim(\mathfrak{g}) \times \dim(\mathfrak{g})}$, also assuming that incoming message $\boldsymbol{m}_{f_i \rightarrow n_j} \sim \mathfrak{N}(\boldsymbol{\mu}_{f_i \rightarrow n_j}, \boldsymbol{\Lambda}_{f_i \rightarrow n_j})$ take the same form. In contrast to conventional vector spaces, the precision matrices are expressed relative to their associated elements in the Lie group, implying that

one must warp them to a privileged frame of reference. An intuitive choice for such a frame is the latest estimate of the node's state, denoted as $\boldsymbol{\mu}_{n_j}^0$ and $\boldsymbol{\Lambda}_{n_j}^0$, resulting in the following transformation rules to warp the messages.

$$\boldsymbol{\tau}_{f_i \to n_j}^0 = \boldsymbol{\mu}_{f_i \to n_j}^0 \boxminus \boldsymbol{\mu}_{n_j}^0 \quad \in \mathbb{R}^{\dim(\mathfrak{g})} \tag{14}$$

$$\boldsymbol{\Lambda}_{f_i \to n_j}^0 = \left[\frac{\partial \boldsymbol{\tau}_{f_i \to n_j}^0}{\partial \boldsymbol{\mu}_{f_i \to n_j}}\right]^\top \boldsymbol{\Lambda}_{f_i \to n_j} \left[\frac{\partial \boldsymbol{\tau}_{f_i \to n_j}^0}{\partial \boldsymbol{\mu}_{f_i \to n_j}}\right] \quad \in \mathbb{R}^{\dim(\mathfrak{g}) \times \dim(\mathfrak{g})} \tag{15}$$

After this conversion, both $\boldsymbol{\tau}_{f_i \to n_j}^0$ and $\boldsymbol{\Lambda}_{f_i \to n_j}^0$ are elements in the tangent (matrix) space relative to the privileged frame and must then be summed in analogy to Eq. (13) to yield the intermediate, incremental values

$$\boldsymbol{\tau}_{n_j}^+ = \alpha_{n_j} \sum_{f_i \in N(n_j)} \boldsymbol{\Lambda}_{f_i \to n_j}^0 \boldsymbol{\tau}_{f_i \to n_j}^0 \quad \text{and} \quad \boldsymbol{\Lambda}_{n_j}^+ = \sum_{f_i \in N(n_j)} \boldsymbol{\Lambda}_{f_i \to n_j}^0, \tag{16}$$

where α_{n_j} denotes an optional step size. The above increments must then again be warped to the updated frame of reference by evaluating

$$\boldsymbol{\mu}_{n_j} = \boldsymbol{\mu}_{n_j}^0 \boxplus \boldsymbol{\tau}_{n_j}^+ \quad \text{and} \quad \boldsymbol{\Lambda}_{n_j} = \left[\frac{\partial \boldsymbol{\mu}_{n_j}}{\partial \boldsymbol{\tau}_{n_j}^+}\right]^\top \boldsymbol{\Lambda}_{n_j}^+ \left[\frac{\partial \boldsymbol{\mu}_{n_j}}{\partial \boldsymbol{\tau}_{n_j}^+}\right]. \tag{17}$$

Note, however, that priors can not be injected as simplistically as in Eq. (13) and need to be treated as proper factors in the context of Lie groups instead.

Node-to-factor Messages: The generation of node-to-factor messages mirrors the expressions from Eq. (16), differing only in excluding information stemming from the target factor in the sums. Hence, they encompass

$$\boldsymbol{\tau}_{n_j \to f_k}^+ = \sum_{f_i \in N(n_j) \setminus f_k} \boldsymbol{\Lambda}_{f_i \to n_j}^0 \boldsymbol{\tau}_{f_i \to n_j}^0 \quad \text{and} \quad \boldsymbol{\Lambda}_{n_j \to f_k}^+ = \sum_{f_i \in N(n_j) \setminus f_k} \boldsymbol{\Lambda}_{f_i \to n_j}^0 \tag{18}$$

alongside their respective linearization point $\boldsymbol{\mu}_{n_j}^0$ to form the outgoing message triplet $(\boldsymbol{\mu}_{n_j}^0, \boldsymbol{\tau}_{n_j \to f_k}^+, \boldsymbol{\Lambda}_{n_j \to f_k}^+)$ destined for the factor f_k. Note, however, that another reevaluation of Eqs. (14) and (15) is required to generate outgoing messages that leverage the latest state estimate from Eq. (17).

Factor Updates: In analogy to previous paragraphs, factors $f_i \in \mathcal{G}$ depend on a collection of connected, neighboring nodes $N(f_i)$ which collectively determine the linearization point $\boldsymbol{\theta}_{f_i}^0$ for the residual evaluation $\bar{\boldsymbol{r}}_{f_i}$ from Eq. (2). The computation of the factor-to-node messages $\boldsymbol{m}_{f_i \to N(f_i)}$ depends on the factor beliefs $B(f_i) = \mathcal{N}^{-1}(\boldsymbol{\eta}_{f_i}^0, \boldsymbol{\Lambda}_{f_i}^0)$ obtained from Eq. (12) as well as the auxiliary, intermediate quantities $\boldsymbol{\eta}_{f_i}'$ and $\boldsymbol{\Lambda}_{f_i}'$ (used in the next paragraph) defined as

$$\boldsymbol{\eta}_{f_i}' = \boldsymbol{\eta}_{f_i}^0 + \boldsymbol{\eta}_{N(f_i) \to f_i}^+ \quad \text{and} \quad \boldsymbol{\Lambda}_{f_i}' = \boldsymbol{\Lambda}_{f_i}^0 + \boldsymbol{\Lambda}_{N(f_i) \to f_i}^+ \tag{19}$$

Above, $\boldsymbol{\eta}_{N(f_i) \to f_i}^+$ and $\boldsymbol{\Lambda}_{N(f_i) \to f_i}^+$ denote stacked vector and block diagonal matrix versions of the neighborhood-to-factor messages (see Eqs. (20) and (21)).

Factor-to-Node Messages: We illustrate the computation of the factor-to-node messages (see Fig. 3b) assuming a factor f_i which depends on two nodes, namely n_a and n_b. Thus, Eq. (19) takes the following form

$$\boldsymbol{\eta}'_{f_i} = \begin{bmatrix} \boldsymbol{\eta}'_a \\ \boldsymbol{\eta}'_b \end{bmatrix} = \boldsymbol{\eta}^0_{f_i} + \begin{bmatrix} \boldsymbol{\eta}^+_{n_a \to f_i} \\ \boldsymbol{\eta}^+_{n_b \to f_i} \end{bmatrix} \tag{20}$$

$$\boldsymbol{\Lambda}'_{f_i} = \begin{bmatrix} \boldsymbol{\Lambda}'_{aa} & \boldsymbol{\Lambda}'^{\top}_{ba} \\ \boldsymbol{\Lambda}'_{ba} & \boldsymbol{\Lambda}'_{bb} \end{bmatrix} = \boldsymbol{\Lambda}^0_{f_i} + \begin{bmatrix} \boldsymbol{\Lambda}^+_{n_a \to f_i} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}^+_{n_b \to f_i} \end{bmatrix}. \tag{21}$$

To retrieve the message $\boldsymbol{m}_{f_i \to n_a} = \mathfrak{N}(\boldsymbol{\mu}_{f_i \to n_a}, \boldsymbol{\Lambda}_{f_i \to n_a})$, the remaining nodes (*i.e.* n_b in this example) must initially be marginalized by computing the Schur complement defined as

$$\boldsymbol{\eta}'_{f_i \to n_a} = \boldsymbol{\eta}^0_a - \boldsymbol{\Lambda}'^{\top}_{ba} \boldsymbol{\Lambda}'^{-1}_{bb} \boldsymbol{\eta}'_b \quad \text{and} \quad \boldsymbol{\Lambda}'_{f_i \to n_a} = \boldsymbol{\Lambda}^0_{aa} - \boldsymbol{\Lambda}'^{\top}_{ba} \boldsymbol{\Lambda}'^{-1}_{bb} \boldsymbol{\Lambda}'_{ba}. \tag{22}$$

In a similar vein, one obtains $\boldsymbol{m}_{f_i \to n_b}$ by permuting $\boldsymbol{\eta}'_{f_i}$ and $\boldsymbol{\Lambda}'_{f_i}$ such that n_a is marginalized instead of n_b. In particular,

$$\begin{bmatrix} \boldsymbol{\eta}'_b \\ \boldsymbol{\eta}'_a \end{bmatrix} = P\boldsymbol{\eta}'_{f_i} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{\Lambda}'_{bb} & \boldsymbol{\Lambda}'^{\top}_{ab} \\ \boldsymbol{\Lambda}'_{ab} & \boldsymbol{\Lambda}'_{aa} \end{bmatrix} = P\boldsymbol{\Lambda}'_{f_i}P^{\top}, \tag{23}$$

which also naturally extends to factors touching more than two nodes. Readers might notice that our formulation differs from others found in the literature; this has the advantage that the formulations from Eqs. (20) and (21) allow for extremely efficient in-place permutations of $\boldsymbol{\eta}'_{f_i}$ and $\boldsymbol{\Lambda}'_{f_i}$ to marginalize many-node factors, limiting reallocations and recomputations. Mirroring the process in Eq. (17), it is necessary to convert the incremental values $\boldsymbol{\eta}'_{f_i \to n_a}$ and $\boldsymbol{\Lambda}'_{f_i \to n_a}$ into the updated, outgoing frame of reference, to obtain the message $\boldsymbol{m}_{f_i \to n_a}$, relying on an optional step size α_{f_i}. That is

$$\boldsymbol{\tau}'_{f_i \to n_a} = \alpha_{f_i} \boldsymbol{\Lambda}'^{-1}_{f_i \to n_a} \boldsymbol{\eta}'_{f_i \to n_a} \in \mathbb{R}^{\dim(\mathfrak{g})}, \quad \boldsymbol{\mu}_{f_i \to n_a} = \boldsymbol{\mu}^0_{n_a} \boxplus \boldsymbol{\tau}'_{f_i \to n_a} \in G \tag{24}$$

$$\text{and} \quad \boldsymbol{\Lambda}_{f_i \to n_a} = \left[\frac{\partial \boldsymbol{\mu}_{f_i \to n_a}}{\partial \boldsymbol{\tau}'_{f_i \to n_a}} \right]^{\top} \boldsymbol{\Lambda}'_{f_i \to n_a} \left[\frac{\partial \boldsymbol{\mu}_{f_i \to n_a}}{\partial \boldsymbol{\tau}'_{f_i \to n_a}} \right] \in \mathbb{R}^{\dim(\mathfrak{g}) \times \dim(\mathfrak{g})}. \tag{25}$$

Robust Residuals and Energies: It is well understood that the presence of outliers in the optimization problem from Eq. (3) causes substantial issues in terms of converging to a globally optimal solution $\boldsymbol{\Theta}^*$ due to the quadratic nature of the occurring cost terms. The same holds in the context of GBP, where outliers are bound to heavily influence the found solution as well. To address this issue we take inspiration from established NLLS approaches [1] and apply robust loss functions ρ to Eqs. (3) and (9), resulting in modified expressions for the robust energies $\breve{E}$ and the robust optimal solution $\breve{\boldsymbol{\Theta}}^*$, provided for completeness only.

$$\breve{E}(t, \boldsymbol{\theta}_s) = \frac{1}{2} \rho\left(\bar{\boldsymbol{r}}^{\top} \bar{\boldsymbol{r}}\right) \tag{26}$$

$$\breve{\boldsymbol{\Theta}}^* = \underset{\boldsymbol{\Theta}}{\mathrm{argmin}} \left[\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \breve{E}(t, \boldsymbol{\theta}_s) \right] \tag{27}$$

In particular, based on the methodology from Triggs *et al.* [41], the robust residual and Jacobian then take the following forms.

$$\alpha^2 - 2\alpha - \frac{2\rho''}{\rho'} \bar{\boldsymbol{r}}_i^\top \bar{\boldsymbol{r}}_i \overset{!}{=} 0 \tag{28}$$

$$\breve{\boldsymbol{r}}_i = \frac{\sqrt{\rho'}}{1 - \alpha} \bar{\boldsymbol{r}}_i \quad \breve{\boldsymbol{J}}_i = \sqrt{\rho'} \left(1 - \alpha \frac{\bar{\boldsymbol{r}}_i \bar{\boldsymbol{r}}_i^\top}{\|\bar{\boldsymbol{r}}_i\|^2} \right) \bar{\boldsymbol{J}}_i \tag{29}$$

Non-Gaussian Nodes: Another extension to conventional GBP lies in classifying Gaussian nodes as either pure constants or as stochastic variables. This distinction streamlines computation given that the message marginalization $\boldsymbol{m}_{n_j \to f_i}$ across nodes *via* Cholesky decomposition exposes complexity $\mathcal{O}(n^3)$ which quickly becomes prohibitive. In particular, allowing non-variable parameters (*e.g.* extrinsic, intrinsic, offset *etc.*) to be treated as pure constants considerably reduces the dimensionality of the marginalization procedure without any drawbacks.

3.4 Sensor Models

In the following, we detail the employed absolute and visual sensor models, where expressions marked with ˆ indicate optimizable parameters. Conversely, unless explicitly mentioned, all other variables are considered known and constant.

Absolute Sensor Model The predictions of absolute sensor measurements $\hat{\boldsymbol{m}}_s(t_m, \Theta_s)$ (*e.g.* absolute pose estimates extracted from AprilTags [29] or measurements obtained from a Motion Capture (MoCap) system) are evaluated using Eq. (30) and subsequently compared against their corresponding true measurements by applying the metric $\boxminus_{\boldsymbol{\mu}}$ in Eq. (31) where we make use of the logarithmic map for elements in $\mathbb{SO}(3)$, and $\mathbb{SU}(2)$ respectively.

$$\hat{\boldsymbol{m}}_s(t_m, \Theta_s) = \hat{\boldsymbol{T}}_{ws}(t_m) = \hat{\boldsymbol{T}}_{wb}(t_m) \, \boldsymbol{T}_{bs} \tag{30}$$

$$\hat{\boldsymbol{T}}_{ws} \boxminus_{\boldsymbol{\mu}} \boldsymbol{T}_{ws} = \begin{bmatrix} \log\left(\hat{\boldsymbol{q}}_{ws} \, \boldsymbol{q}_{ws}^{-1}\right) \\ \hat{\boldsymbol{t}}_{ws} - \boldsymbol{t}_{ws} \end{bmatrix} \in \mathbb{R}^6 \tag{31}$$

Visual Sensor Model Abstracting from the specific camera parameters, such as its intrinsics and its distortion model, we introduce the mapping $\pi(\cdot)$ in Eq. (32) to infer individual projections of landmarks l_w onto the image plane at time t_m. For visual, pixel-based measurements, the metric $\boxminus_{\boldsymbol{\mu}}$ is chosen to be equivalent to trivial Euclidean subtraction.

$$\hat{\boldsymbol{m}}_s(t_m, \Theta_s) = \hat{\boldsymbol{p}}_s(t_m) = \pi(\boldsymbol{T}_{sb} \hat{\boldsymbol{T}}_{bw}(t_m), \, l_w) \tag{32}$$

$$\hat{\boldsymbol{p}}_s \boxminus_{\boldsymbol{\mu}} \boldsymbol{p}_s = \hat{\boldsymbol{p}}_s - \boldsymbol{p}_s \in \mathbb{R}^2 \tag{33}$$

3.5   System

Overview Our system is designed to efficiently support both batch and windowed optimizations, crucial for applications such as Bundle Adjustment (BA) and SLAM. Inspired by Ceres [1], it allows dynamic modification of nodes and factors between solver calls, features multiple solving strategies, namely synchronous and dropout, determining the order of node and factor updates and leverages multi-threading to exploit the inherent parallelism of GBP methods, further boosting the overall performance. Optimizable parameters are added and removed in analogy to Ceres [1], with the notable distinction that GBP-based methods necessitate both an initial mean and covariance estimate for each parameter, unlike standard NLLS methods. This requirement, however, poses no practical hindrance as conservative guesses on the covariance suffice to bootstrap the algorithm.

Symbolic Factors Targeting the long-standing challenge of obtaining efficient, analytical expressions and derivatives of spline-based motion parameterizations and their associated cost factors, here, we leverage SymForce [22], a symbolic code generation framework, to completely automate this cumbersome, time-intensive and error-prone process. Taking inspiration from Sommer *et al.* [40], who exploited Lie-group-specific properties to simplify spline-related mathematical expressions, we supercharge their recursive spline formulation by combining it with SymForce [22] to obtain ultra-efficient C++ cost factor implementations. Furthermore, recognizing the broader relevance of automated factor generation for (continuous-time) robotics applications, our framework offers a comprehensive library featuring spline-based and standard factors designed to seamlessly interoperate with both our framework and Ceres [1], facilitating widespread adoption.

Update Strategies Amongst many possible update strategies, we focus our attention on synchronous and dropout updates. Synchronous updates closely follow traditional NLLS algorithms, where residuals are completely re-evaluated in every solver iteration and where (non-constant) nodes and factors are sequentially updated in an alternating fashion to ensure optimal convergence. In contrast to traditional approaches, GBP, however, also accommodates selective updates based on a dropout strategy. That is, one assigns probabilities, d_n for nodes and d_f for factors, dictating their likelihood of being updated in each iteration. Although this may slow convergence, it intentionally compensates for stability challenges in loopy graphs while mimicking real-world scenarios with imperfect communication channels and delayed messages.

4   Experiments

We evaluate the proposed system, coined Hyperion, empirically through simulations in MoCap (absolute) and localization settings, using a temporal interval of 0.1 seconds between adjacent spline bases along with the empirical step sizes

$\alpha_{n_j} = \alpha_{f_i} = 0.7$ across all cases. The simulated trajectories span 10 seconds and mimic real-world conditions with appropriate initial perturbations, sensor acquisition rates, and measurement noise levels. Our analysis focuses on benchmarking Hyperion against Ceres [1], an established, centralized NLLS solver, and providing in-depth ablation studies on different aspects of the framework itself.

4.1   Absolute Setup

We commence our analysis with the absolute sensor configuration, capturing measurements akin to the ones stemming from a MoCap system, where individual factors model direct observations of the underlying motion without auxiliary parameters/nodes. As such, they embody the most fundamental category of constraints, suggesting that derived factors, such as landmark projections or relative measurements, merely extend these elements, rendering them ideal candidates to analyze key properties of the proposed approach. In the following, we assume an acquisition rate of 40 Hz matching the real-world specifications.

Our initial examination centers on comparing the convergence behavior of Hyperion against Ceres [1] across varying levels of initial perturbations and measurement noise. To this end, both solvers are identically initialized to a modified ground truth motion, encompassing perturbations in rotations and translations. Furthermore, we make use of the synchronized vertex updates if not stated otherwise, aiming for a fair comparison against Ceres.

The qualitative outcomes of this setup are illustrated in Fig. 1, demonstrating that both Hyperion and Ceres converge to identical solutions even under substantial initial perturbation and considerable measurement noise. These observations are quantified in Tabs. 1a and 1b, indicating that motion estimates obtained from Hyperion and Ceres closely align across all levels of perturbation and noise.

Message dropouts, mimicking imperfect communication, have also been shown to enhance convergence [27, 31] in loopy graphs. Hence, we study their effect on the proposed GBP framework, summarizing key insights in Fig. 4a, revealing that Hyperion consistently converges toward identical solutions across various dropout ratios. Despite this, they also influence the required iteration count (*i.e.*, updates of all vertices), with every 10% increase in dropouts entailing an additional 2-4 iterations until convergence is reached.

Furthermore, we find that different motion parameterizations, specifically cubic Z- and B-Splines [4, 17], also influence the convergence of the proposed approach, which we summarize in Fig. 7b. A notable difference between the traditional and our proposed solver lies in how motion parameterizations affect the convergence speed of the latter. Our findings suggest that Z-Splines, which are interpolating rather than approximating splines, typically yield better-conditioned solutions since their bases must lie on the motion estimate itself, implicitly constraining the space of possible solutions. Naturally, the centralized NLLS solver sets a baseline for the required iteration count to resolve an optimization problem, serving as a comparative standard for our method. Thus, while GBP-based approaches are inherently distributed, Fig. 7b indicates that they impose between 2 to 4 additional iterations to achieve convergence.

Perturbation [m/rad]						
	1e-5	1e-4	1e-3	1e-2	1e-1	1e-0
Ours						
R [rad]	5.2e-6	5.2e-6	5.2e-6	5.2e-6	5.9e-6	5.3e-6
t [m]	5.8e-6	5.8e-6	5.9e-6	5.9e-6	6.0e-6	1.3e-5
Ceres						
R [rad]	5.2e-6	5.2e-6	5.2e-6	5.2e-6	5.2e-6	5.2e-6
t [m]	5.9e-6	5.9e-6	5.9e-6	5.9e-6	5.9e-6	5.9e-6

(a) Perturbation Tolerance

Noise [m/rad]						
	1e-5	1e-4	1e-3	1e-2	1e-1	1e-0
Ours						
R [rad]	5.2e-6	5.2e-5	5.2e-4	5.2e-3	5.3e-2	5.6e-1
t [m]	5.8e-6	5.9e-5	5.9e-4	5.9e-3	5.9e-2	5.9e-1
Ceres						
R [rad]	5.2e-6	5.2e-5	5.2e-4	5.2e-3	5.2e-2	5.5e-1
t [m]	5.9e-6	5.9e-5	5.9e-4	5.9e-3	5.9e-2	5.9e-1

(b) Noise Tolerance

Table 1: Root Mean Square Errors (RMSEs) in rotation (R) and translation (t) resulting from Ceres [1] and Hyperion under different perturbation levels (a), different noise levels (b) respectively. The solvers run to convergence or terminate after 50 iterations. The perturbation/noise setup assumes the lowest noise/perturbation, respectively.
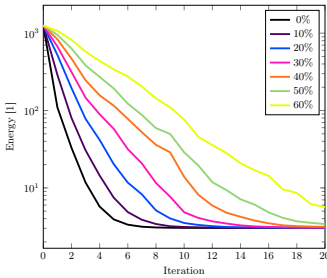
(a) Convergence: Dropout

Setup			Pose [s]		Velocity [s]		Acceleration [s]		Avg.
$\mathcal{L}$	k	$\partial/\partial\mathcal{B}$	Ours	Basalt	Ours	Basalt	Ours	Basalt	Speedup
$\mathbb{SO}(3)$	4	✗	**1.64e-7**	3.16e-7	**9.49e-8**	2.90e-7	**1.12e-7**	3.28e-7	2.64x
$\mathbb{SO}(3)$	4	✓	**5.03e-7**	6.70e-6	**4.05e-7**	7.82e-6	**5.11e-7**	9.46e-6	17.05x
$\mathbb{SO}(3)$	5	✗	**1.96e-7**	4.14e-7	**1.32e-7**	3.67e-7	**1.39e-7**	3.96e-7	2.58x
$\mathbb{SO}(3)$	5	✓	**6.89e-7**	1.08e-5	**5.82e-7**	1.27e-5	**7.87e-7**	1.58e-5	19.19x
$\mathbb{SO}(3)$	6	✗	**2.17e-7**	4.78e-7	**1.82e-7**	4.42e-7	**1.81e-7**	4.83e-7	2.43x
$\mathbb{SO}(3)$	6	✓	**8.19e-7**	1.57e-5	**7.51e-7**	1.87e-5	**1.01e-6**	2.43e-5	22.71x
$\mathbb{SE}(3)$	4	✗	**1.62e-7**	7.03e-7	**1.38e-7**	7.46e-7	**1.34e-7**	6.88e-7	4.96x
$\mathbb{SE}(3)$	4	✓	**5.71e-7**	4.69e-5	**9.11e-7**	5.35e-5	**1.12e-6**	6.25e-5	65.56x
$\mathbb{SE}(3)$	5	✗	**1.96e-7**	7.38e-7	**1.91e-7**	8.73e-7	**1.70e-7**	9.15e-7	4.57x
$\mathbb{SE}(3)$	5	✓	**7.32e-7**	9.64e-5	**1.27e-6**	9.44e-5	**1.47e-6**	1.14e-4	94.53x
$\mathbb{SE}(3)$	6	✗	**2.53e-7**	9.11e-7	**2.34e-7**	1.12e-6	**2.23e-7**	1.12e-6	4.47x
$\mathbb{SE}(3)$	6	✓	**9.29e-7**	1.25e-4	**1.54e-6**	1.67e-4	**1.99e-6**	1.75e-4	110.31x

(b) B-Spline Evaluation Timings

Fig. 4: (a) Graph energy vs. number of iterations conditioned on the dropout probability in the absolute setup (batch). (b) Performance comparison between our symbolically, auto-generated and optimized B-Spline implementation and the recursively-defined, hand-crafted implementation used by Sommer *et al.* [40] on an M3 Max @4.05GHz.

Lastly, we benchmark Hyperion's real-world performance using a handheld camera to track a ChArUco [14] board, capturing a 60-second trajectory at 30 Hz with an iPhone 13 Pro. The analysis, detailed in Fig. 5, closely mirrors our previous analysis. However, we find that real-world motions, which innately expose a greater level of volatility than simulated ones, expose slower convergence.

4.2   Localization Setup

In this section, we analyze our approach in a localization setup, where we model an image acquisition rate of 20 Hz and 50 randomized, observable landmarks at a range of 2 to 6 m across the modeled motion, mimicking real-world conditions.

Overall, the results from the localization setup are consistent with our earlier observations, as detailed in Figs. 6, 7a and 7c. However, Fig. 7a indicates that this setup is more agnostic to lower dropout ratios compared to the absolute case. Furthermore, the disparity between different motion parameterizations becomes more pronounced over the previous setup as may be observed from
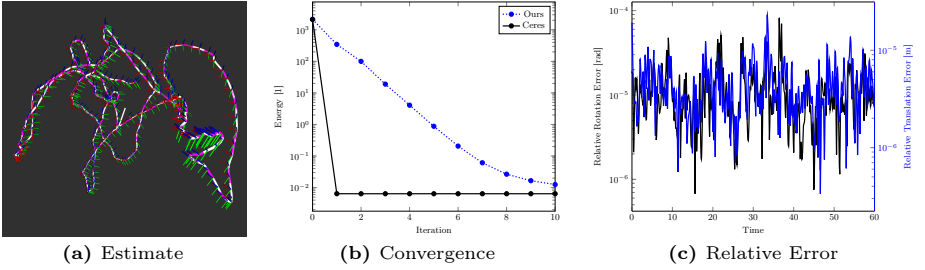
(a) Estimate   **(b)** Convergence   **(c)** Relative Error

Fig. 5: ChArUco [14] setup with overlapping motion estimates (a) from Hyperion and Ceres [1] in magenta and white, respectively. Illustration of the corresponding convergence (b) and the relative errors between the two converged estimates (c).
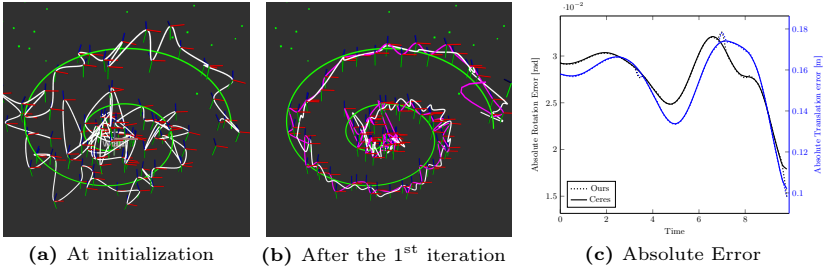

(a) At initialization   **(b)** After the 1$^{\text{st}}$ iteration   **(c)** Absolute Error

Fig. 6: The motion estimates for Hyperion and a conventional NLLS solver [1] in magenta and white, respectively, converge to similar solutions close to ground truth (in green) in the localization setup, demonstrating robust convergence under poor pose (± 0.20 m/rad) and landmark initialization (± 0.20 m) with ± 1 px measurement noise.

Fig. 7c. In general, visual setups also display higher volatility over absolute ones, with notable differences in estimates, especially towards the head and tails of the optimized splines. In a similar vein, we found that these numerical instabilities tend to have a destabilizing effect in loop graphs, posing considerable challenges to deploying the proposed system as a comprehensive SLAM system at present.

4.3   Performance Analysis

As highlighted in Sec. 3.5, our framework leverages SymForce [22] to automate the generation of analytic cost factors, aiming to outperform hand-crafted implementations while mitigating development effort and programmatic errors. Thus, we test our auto-generated B-Spline implementation against Sommer *et al.*'s optimized, hand-crafted version [40]. The comparison, detailed in Fig. 4b, assesses the timings of pose, velocity, and acceleration evaluations for B-Splines of order k, within a Lie group $\mathcal{L}$, potentially including the Jacobians $\partial/\partial \mathcal{B}$ with respect to the bases $\mathcal{B}_i$. Figure 4b shows that our automated spline implementations significantly outperform the manual ones from [40], with speedups ranging from 2.43x to 110.31x. The performance gains are particularly notable in derivative
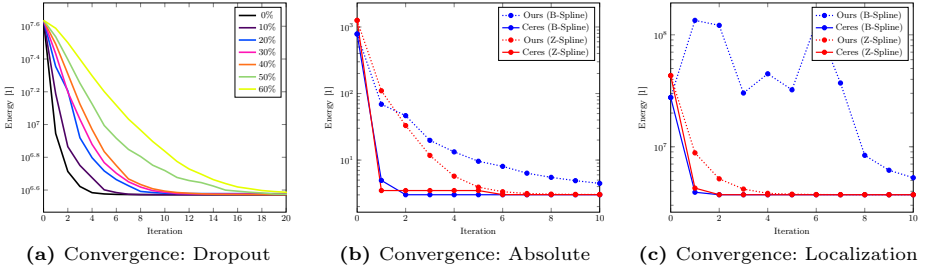
(a) Convergence: Dropout **(b)** Convergence: Absolute **(c)** Convergence: Localization

Fig. 7: (a) Graph energy vs. number of iterations conditioned on the dropout probability in the localization setup. Convergence comparison of different splines and solver variants in the absolute setups (b) and the localization setup (c), respectively.

evaluations, where [40] employs automatic differentiation to compute the Jacobians. Considering the long-standing computational challenges of CTSLAM methods, our automated approach not only reduces complexity but also narrows the performance divide with discrete-time SLAM.

We further analyze the single-core performance of the proposed GBP solver with Ceres [1], a conventional NLLS solver, within the previously discussed scenarios. For the absolute setup (400 pose measurements), Ceres averages 0.70 ms per iteration, whereas our method averages 4.18 ms to update all graph vertices (*i.e.* about 6x slower). In the localization scenario (10,000 landmark reprojections), Ceres requires 15 ms per iteration, compared to our solver's 112 ms per iteration (*i.e.* 7.5x slower). This performance gap is, however, largely mitigated by considering that Ceres excludes the (implicit) estimation of covariances, significantly reducing computational load, and that it has been extensively refined by numerous contributors over the years. Moreover, the intrinsic distributedness and parallelizability of GBP along with potential enhancements from adaptive vertex updates, leveraging covariance estimates to trigger selective updates, are promising avenues for further performance improvements. The above analysis suggests that, even in its present state, the proposed framework is capable of real-time execution for moderately sized problems.

5   Conclusions

In this work, we present a fast, versatile GBP framework that targets distributed, continuous-time SLAM applications and leverages message-passing algorithms to achieve probabilistic inference. We demonstrate the efficacy and competitiveness of our method against a conventional NLLS solver [1] achieving similar convergence and performance properties in practical settings. We further provide a comprehensive library of high-performance implementations for continuous-time SLAM, comprising motion parameterizations as well as common factors. Based on our experiments, the proposed framework shows great promise in paving the way towards resilient, distributed, continuous-time SLAM solutions.

Acknowledgments

References

1. Agarwal, S., Mierle, K.: Ceres Solver, https://github.com/ceres-solver/ceres-solver, (2023, November 28)
2. Anderson, S., Dellaert, F., Barfoot, T.D.: A Hierarchical Wavelet Decomposition for Continuous-Time SLAM. In: IEEE Int. Conf. Rob. Aut. pp. 373–380 (2014)
3. Bänninger, P., Alzugaray, I., Karrer, M., Chli, M.: Cross-agent Relocalization for Decentralized Collaborative SLAM. In: IEEE Int. Conf. Rob. Aut. pp. 5551–5557 (2023)
4. Becerra-Sagredo, J.T.: Z-Splines: Moment-conserving Cardinal Spline Interpolation of Compact Support for Arbitrarily Spaced Data. Sem. Apl. Math. (2003)
5. Bloesch, M., Burri, M., Omari, S., Hutter, M., Siegwart, R.: Iterated Extended Kalman Filter-based Visual-Inertial Odometry using Direct Photometric Feedback. Int. J. Rob. Reas. **36**(10), 1053–1072 (2017)
6. Bloesch, M., Omari, S., Hutter, M., Siegwart, R.: Robust Visual-Inertial Odometry using a Direct EKF-based Approach. In: IEEE Int. Conf. Int. Rob. Sys. pp. 298–304 (2015)
7. Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M., Tardós, J.D.: ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. IEEE Trans. Rob. **37**(6), 1874–1890 (2021)
8. Davison, A.J., Ortiz, J.: FutureMapping 2: Gaussian Belief Propagation for Spatial AI. ArXiv (2019)
9. Droeschel, D., Behnke, S.: Efficient Continuous-Time SLAM for 3D Lidar-based Online Mapping. In: IEEE Int. Conf. Rob. Aut. pp. 1–9 (05 2018)
10. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale Direct Monocular SLAM. In: Eur. Conf. Comput. Vis. pp. 834–849 (2014)
11. Forster, C., Zhang, Z., Gassner, M., Werlberger, M., Scaramuzza, D.: SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems. IEEE Trans. Rob. **33**(2), 249–265 (2017)
12. Furgale, P., D. Barfoot, T., Sibley, G.: Continuous-Time Batch Estimation using Temporal Basis Functions. Rob. Sc. Sys. **34**, 2088–2095 (05 2012)
13. Furgale, P., Rehder, J., Siegwart, R.: Unified Temporal and Spatial Calibration for Multi-sensor Systems. In: IEEE Int. Conf. Int. Rob. Sys. pp. 1280–1286 (2013)
14. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., Marín-Jiménez, M.J.: Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion. Pattern Recognition **47**(6), 2280–2292 (2014)
15. Hug, D., Bänninger, P., Alzugaray, I., Chli, M.: Continuous-Time Stereo-Inertial Odometry. IEEE Rob. Aut. Letters **7**(3), 6455–6462 (2022)
16. Hug, D., Chli, M.: HyperSLAM: A Generic and Modular Approach to Sensor Fusion and Simultaneous Localization and Mapping in Continuous-Time. In: Int. Conf. 3D Vis. pp. 978–986 (2020)
17. Kaihuai Qin: General Matrix Representations for B-splines. In: Pac. Conf. Comput. Graph. Appl. pp. 37–43 (10 1998)

18. Karrer, M., Chli, M.: Distributed Variable-baseline Stereo SLAM from two UAVs. In: IEEE Int. Conf. Rob. Aut. pp. 82–88 (2021)
19. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based Visual–inertial Odometry using Nonlinear Optimization. Int. J. Rob. Reas. **34**, 314 – 334 (2015)
20. Lovegrove, S., Patron-Perez, A., Sibley, G.: Spline Fusion: A continuous-Time Representation for Visual-Inertial Fusion with Application to Rolling Shutter Cameras. In: Brit. Mach. Vis. Conf. vol. 2, p. 8 (2013)
21. Lynen, S., Achtelik, M.W., Weiss, S., Chli, M., Siegwart, R.: A Robust and Modular Multi-sensor Fusion Approach Applied to MAV Navigation. In: IEEE Int. Conf. Int. Rob. Sys. pp. 3923–3929 (2013)
22. Martiros, H., Miller, A., Bucki, N., Solliday, B., Kennedy, R., Zhu, J., Dang, T., Pattison, D., Zheng, H., Tomic, T., Henry, P., Cross, G., VanderMey, J., Sun, A., Wang, S., Holtz, K.: SymForce: Symbolic Computation and Code Generation for Robotics. In: Rob. Sc. Sys. (2022)
23. Mascaro, R., Teixeira, L., Hinzmann, T., Siegwart, R., Chli, M.: GOMSF: Graph-Optimization Based Multi-Sensor Fusion for robust UAV Pose Estimation. In: IEEE Int. Conf. Rob. Aut. pp. 1421–1428 (2018)
24. Milford, M.J., Wyeth, G.F.: SeqSLAM: Visual Route-based Navigation for Sunny Summer Days and Stormy Winter Nights. In: IEEE Int. Conf. Rob. Aut. pp. 1643–1649 (2012)
25. Mueggler, E., Gallego, G., Rebecq, H., Scaramuzza, D.: Continuous-Time Visual-Inertial Odometry for Event Cameras. IEEE Trans. Rob. **34**(6), 1425–1440 (2018)
26. Mur-Artal, R., Montiel, J., Tardós, J.D.: ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Trans. Rob. **31**, 1147–1163 (2015)
27. Murai, R., Alzugaray, I., Kelly, P.H., Davison, A.J.: Distributed Simultaneous Localisation and Auto-Calibration Using Gaussian Belief Propagation. IEEE Rob. Aut. Letters (2024)
28. Murai, R., Ortiz, J., Saeedi, S., Kelly, P.H., Davison, A.J.: A Robot Web for Distributed Many-Device Localisation. IEEE Trans. Rob. (2023)
29. Olson, E.: AprilTag: A Robust and Flexible Visual Fiducial System. In: IEEE Int. Conf. Rob. Aut. pp. 3400–3407 (2011)
30. Ortiz, J., Evans, T., Davison, A.J.: A Visual Introduction to Gaussian Belief Propagation. ArXiv (2021)
31. Ortiz, J., Evans, T., Sucar, E., Davison, A.J.: Incremental Abstraction in Distributed Probabilistic SLAM Graphs. In: IEEE Int. Conf. Rob. Aut. pp. 7566–7572 (2022)
32. Ortiz, J., Pupilli, M., Leutenegger, S., Davison, A.J.: Bundle Adjustment on a Graph Processor. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 2416–2425 (2020)
33. Oth, L., Furgale, P., Kneip, L., Siegwart, R.: Rolling Shutter Camera Calibration. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1360–1367 (2013)
34. Peng, Z., Xu, Y., Yan, M., Yin, W.: ARock: An Algorithmic Framework for Asynchronous Parallel Coordinate Updates. J. Sc. Comput. **38**(5), A2851–A2879 (2016)
35. Qin, T., Cao, S., Pan, J., Shen, S.: A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors. ArXiv (2019)
36. Qin, T., Li, P., Shen, S.: VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. IEEE Trans. Rob. **34**(4), 1004–1020 (2018)
37. Qin, T., Pan, J., Cao, S., Shen, S.: A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors. ArXiv (2019)

38. Schmuck, P., Chli, M.: CCM-SLAM: Robust and Efficient Centralized Collaborative Monocular Simultaneous Localization and Mapping for Robotic Teams. J. Fld. Rob. **36**(4), 763–781 (2019)
39. Schubert, D., Demmel, N., Usenko, V., Stuckler, J., Cremers, D.: Direct Sparse Odometry With Rolling Shutter. In: Eur. Conf. Comput. Vis. pp. 682–697 (2018)
40. Sommer, C., Usenko, V., Schubert, D., Demmel, N., Cremers, D.: Efficient Derivative Computation for Cumulative B-Splines on Lie Groups. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 11145–11153 (2020)
41. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle Adjustment – A Modern Synthesis. In: Vision Algorithms: Theory and Practice. pp. 298–372 (2000)
42. Usenko, V., Demmel, N., Schubert, D., Stückler, J., Cremers, D.: Visual-Inertial Mapping With Non-linear Factor Recovery. IEEE Rob. Aut. Letters **5**(2), 422–429 (2019)
43. Wang, R., Schworer, M., Cremers, D.: Stereo DSO: Large-scale Direct Sparse Visual Odometry with Stereo Cameras. In: Int. Conf. Comput. Vis. pp. 3903–3911 (2017)
44. Wang, S., Clark, R., Wen, H., Trigoni, N.: DeepVO: Towards end-to-end Visual Odometry with Deep Recurrent Convolutional Neural Networks. In: IEEE Int. Conf. Rob. Aut. pp. 2043–2050 (2017)
45. Wang, X., Xue, F., Yan, Z., Dong, W., Wang, Q., Zha, H.: Continuous-Time Stereo Visual Odometry Based on Dynamics Model. In: Asian Conf. Comput. Vis. pp. 388–403 (2018)
46. Yang, A.J., Cui, C., Bârsan, I.A., Urtasun, R., Wang, S.: Asynchronous Multi-view SLAM. In: IEEE Int. Conf. Rob. Aut. pp. 5669–5676 (2021)