G-solver: Gaussian Belief Propagation and Gaussian Processes for Continuous-Time SLAM

Davide Ceriola Simone Ferrari Luca Di Giammarino Leonardo Brizi Giorgio Grisetti
Sapienza University of Rome, Italy
{ceriola,ferrari,digiammarino,brizi,grisetti}@diag.uniroma1.it

1. Introduction

Accurate state estimation is central to robotics and vision tasks such as Simultaneous Localization and Mapping (SLAM), mapping, and navigation. In real-world deployments, systems must fuse diverse sensors—cameras, Inertial Measurement Unit (IMU), GNSS, and Light Detection And Ranging (LiDAR)—each with complementary strengths and weaknesses. Robustness to sensor diversity and domain shifts (e.g., varying environments, lighting, or motion patterns) is therefore essential for scalability beyond controlled settings. Traditional approaches often adopt a discrete-time formulation, solved via Non Linear Least Squares (NLLS) optimization [1, 13]. While effective, these methods require precise synchronization, cannot easily provide estimates at intermediate times, and scale poorly with high-frequency or long-duration data. Continuous-time formulations address these issues by modeling trajectories as smooth functions. B-splines [8] are widely used for this purpose, but they require careful knot placement and do not naturally capture uncertainty, which limits adaptability across sensing conditions. Gaussian Processs (GPs) [5, 16] offer a principled alternative, representing trajectories as distributions over functions with smooth motion priors and uncertainty-aware inference. This makes them attractive for integrating heterogeneous data streams and handling domain shifts. However, most GP-based methods rely on batch optimization, which restricts real-time operation at scale. Distributed inference with Gaussian Belief Propagation (GBP) provides a compelling solution [6, 15]. By updating estimates incrementally through local message passing on factor graphs, GBP enables scalable and decentralized inference, well suited for continuous sensor streams and large environments. Recent systems such as Hyperion [10] demonstrate this potential using spline-based trajectories. In this paper, we present G-solver, a general-purpose framework that integrates GP priors with GBP for continuous-time state estimation (Fig. 1). Our approach preserves the smoothness and uncertainty modeling of GPs while enabling scalable distributed inference. This makes it particularly effective in settings with heterogeneous sensors and shifting domains. Experiments confirm that G-solver achieves higher accuracy than existing methods while maintaining competitive runtime. Our contributions are:

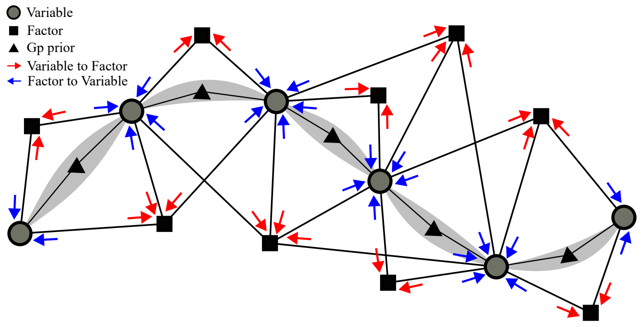
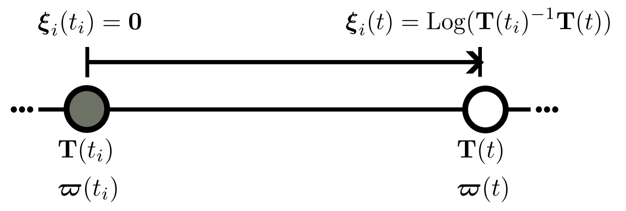


Figure 1. **G-solver**. An illustration of a factor graph with the message-passing algorithm in action, where arrows indicate the GBP messages exchanged between nodes. The state variable is modeled continuously using GP prior factors, and the associated GP covariance is depicted in gray.

- a novel formulation combining GP priors with GBP for continuous-time SLAM;
- efficient and decentralized inference that naturally handles asynchronous and diverse sensor data;
- a consistent representation where GP parameters jointly govern optimization and trajectory inference;
- an open-source implementation of G-solver.

2. G-solver

G-solver is a continuous-time SLAM solver that combines GP priors with GBP. Motion between poses is modeled with a constant-velocity GP prior, introduced as a binary factor in the graph. Compared to spline-based methods [10], this formulation requires only one tunable hyperparameter, naturally encodes uncertainty (Fig. 3), and directly contributes to the optimization. Since states lie on $\mathbb{SE}(3)$, we rely on $\boxplus/\boxminus$ operators [9] to move between the manifold $\mathcal{M}$ and tangent space $\mathrm{T}_{\mu}\mathcal{M}$, enabling linearization and uncertainty



$\boldsymbol{\xi}_i(t_i) = \mathbf{0}$
$\boldsymbol{\xi}_i(t) = \mathrm{Log}(\mathbf{T}(t_i)^{-1}\mathbf{T}(t))$
$\mathbf{T}(t_i)$
$\boldsymbol{\varpi}(t_i)$
$\mathbf{T}(t)$
$\boldsymbol{\varpi}(t)$

Figure 2. **Locally Linear Approximation.** Between times t_i and t_{i+1}, a constant-velocity GP prior links local pose variables $\boldsymbol{\xi}_i(t)$ with the global trajectory state $\{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}$.

propagation. Following standard factor graph notation [7], variables x_i and factors f_i are represented as Gaussians:

$$x_i \sim \mathcal{N}^{-1}(\boldsymbol{\eta}_{x_i}, \boldsymbol{\Lambda}_{x_i}), \quad f_j \sim \mathcal{N}^{-1}(\boldsymbol{\eta}_{f_j}, \boldsymbol{\Lambda}_{f_j}), \tag{1}$$

with mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1}$, and information vector $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$. Inference is then carried out by GBP local message passing.

2.1. Continuous-time estimation using GP

To obtain a continuous-time estimate, the trajectory is represented as a GP $\mathbf{x}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t,t'))$ with the Markov state

$$\mathbf{x}(t) \doteq \{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}, \tag{2}$$

where $\mathbf{T}(t) \in \mathbb{SE}(3)$ is the pose and $\boldsymbol{\varpi}(t) \in \mathbb{R}^6$ the body-centric velocity ($[\boldsymbol{\nu}(t), \boldsymbol{\omega}(t)]^\top$). We incorporate the above in the factor graph by defining a constant-velocity prior encoded as a binary factor between consecutive poses (Fig. 2), yielding the error term:

$$\mathbf{e}_i = \begin{bmatrix} (\mathbf{T}_{i+1} \boxminus \mathbf{T}_i) \boxminus \mathrm{Exp}(\Delta t_i \boldsymbol{\varpi}_i) \\ \begin{pmatrix} {}^i\mathbf{R}_{i+1} & \mathbf{0} \\ \mathbf{0} & {}^i\mathbf{R}_{i+1} \end{pmatrix} \boldsymbol{\varpi}_{i+1} - \boldsymbol{\varpi}_i \end{bmatrix}, \tag{3}$$

which penalizes deviations from constant velocity on the manifold, where $\mathbf{T}_i, \mathbf{T}_{i+1}, \boldsymbol{\varpi}_i, \boldsymbol{\varpi}_{i+1}$ are the *global* state variables, and $\Delta t_i = t_{i+1} - t_i$ is the time difference between consecutive poses. Moving to the *local* Markov state:

$$\boldsymbol{\gamma}_i(t) \doteq \begin{bmatrix} \boldsymbol{\xi}_i(t) \\ \dot{\boldsymbol{\xi}}_i(t) \end{bmatrix}, \quad \boldsymbol{\xi}_i(t) = \mathrm{Log}(\mathbf{T}_i^{-1}\mathbf{T}(t)), \tag{4}$$

where we represent the local pose variable in the tangent space with $\boldsymbol{\xi}_i(t)$, and the velocity expressed in the local frame as:

$$\dot{\boldsymbol{\xi}}_i(t) = \begin{pmatrix} {}^i\mathbf{R}_t & \mathbf{0} \\ \mathbf{0} & {}^i\mathbf{R}_t \end{pmatrix} \boldsymbol{\varpi}(t). \tag{5}$$

As in [2], state transitions follow the constant-velocity model

$$\boldsymbol{\gamma}_i(t_{i+1}) = \boldsymbol{\Phi}(\Delta t_i)\boldsymbol{\gamma}_i(t_i), \quad \boldsymbol{\Phi}(\Delta t_i) = \begin{bmatrix} \mathbf{1} & \Delta t_i \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \tag{6}$$

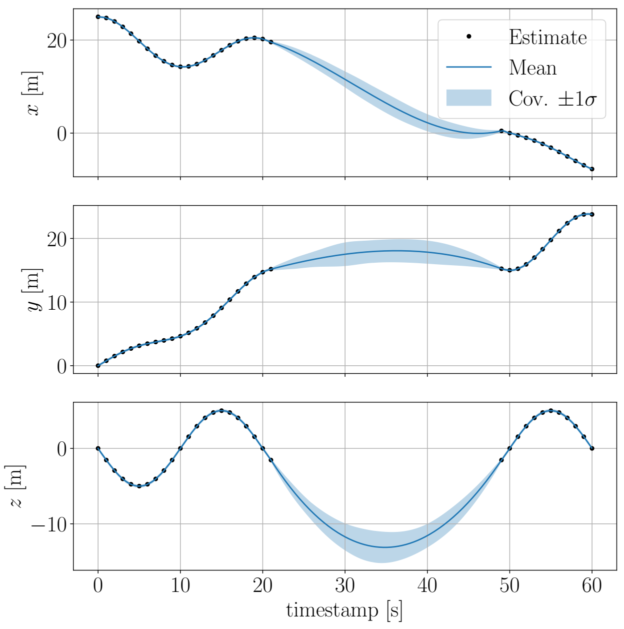


Figure 3. **Trajectory Mean and Covariance.** Continuous-time estimates on a torus trajectory. Uncertainty grows in regions without measurements.

where $\boldsymbol{\Phi}$ is the transition matrix. $\dot{\boldsymbol{\varpi}}(t)$ is modeled as a zero-mean white-noise GP:

$$\dot{\boldsymbol{\varpi}}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C \delta(t-t')) \tag{7}$$

where $\mathbf{Q}_C$ is the power spectral density and the sole hyperparameter of our framework [5]. By integrating this process twice, we obtain the prior covariance matrix:

$$\mathbf{Q}_i = \begin{bmatrix} \frac{1}{3}\Delta t_i^3 \mathbf{Q}_C & \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C \\ \frac{1}{2}\Delta t_i^2 \mathbf{Q}_C & \Delta t_i \mathbf{Q}_C \end{bmatrix}, \tag{8}$$

Finally, each GP factor introduces an energy cost

$$E_i = \tfrac{1}{2}\mathbf{e}_i^\top \mathbf{Q}_i^{-1} \mathbf{e}_i. \tag{9}$$

2.2. Gaussian Belief Propagation with GP

GBP is an iterative message-passing algorithm [11]. In our scenarios, since states lie on $\mathbb{SE}(3)$, incoming messages ($m_{in} \sim \mathcal{N}(\boldsymbol{\mu}_{in}, \boldsymbol{\Lambda}_{in}^{-1})$) must be projected to the tangent space ($m_{in}^{\boldsymbol{\tau}} \sim \mathcal{N}(\boldsymbol{\tau}_{in}, (\boldsymbol{\Lambda}_{in}^{\boldsymbol{\tau}})^{-1})$) as follows:

$$\boldsymbol{\tau}_{in} = \boldsymbol{\mu}_{in} \boxminus \boldsymbol{\mu}_0, \tag{10}$$
$$\boldsymbol{\Lambda}_{in}^{\boldsymbol{\tau}} \approx \boldsymbol{\Lambda}_{in}, \tag{11}$$

where, without loss of stability, we adopt a direct covariance approximation for efficiency, differently from first-order corrections in [10, 15]. Internal computations are then carried out on $\boldsymbol{\tau}_{in} = (\boldsymbol{\xi}_i(t)^\top, \dot{\boldsymbol{\xi}}_i(t)^\top)^\top \in \mathbb{R}^{12}$, a formulation that extends naturally to Euclidean entities (e.g., landmarks in $\mathbb{R}^3$). GBP involves variable updates, factor updates, variable-to-factor message generation, and factor-to-variable message generation. The first three steps follow [15]; we highlight the factor-to-variable message, which differs from [10].

Factor-to-Variable Message A factor-to-variable message represents the probability distribution of the recipient variable that minimizes the factor cost. To compute it, we first condition the factor with incoming messages to obtain the joint distribution over connected variables, and then marginalize out the recipient. Denoting by α the recipient and by β the remaining variables information, we write:

$$\boldsymbol{\eta}_{C_f} = \begin{pmatrix} \boldsymbol{\eta}_\alpha \\ \boldsymbol{\eta}_\beta \end{pmatrix} = \begin{pmatrix} \boldsymbol{\eta}_{f_\alpha} \\ \boldsymbol{\eta}_{f_\beta} + \boldsymbol{\eta}_{x_\beta \to f_j} \end{pmatrix}, \tag{12}$$

$$\boldsymbol{\Lambda}_{C_f} = \begin{bmatrix} \boldsymbol{\Lambda}_{\alpha\alpha} & \boldsymbol{\Lambda}_{\alpha\beta} \\ \boldsymbol{\Lambda}_{\beta\alpha} & \boldsymbol{\Lambda}_{\beta\beta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Lambda}_{f_{\alpha\alpha}} & \boldsymbol{\Lambda}_{f_{\alpha\beta}} \\ \boldsymbol{\Lambda}_{f_{\beta\alpha}} & \boldsymbol{\Lambda}_{f_{\beta\beta}} + \boldsymbol{\Lambda}_{x_\beta \to f_j} \end{bmatrix}, \tag{13}$$

where $\mathcal{N}^{-1}(\boldsymbol{\eta}_f, \boldsymbol{\Lambda}_f)$ is the factor distribution and $\mathcal{N}^{-1}(\boldsymbol{\eta}_{C_f}, \boldsymbol{\Lambda}_{C_f})$ the conditioned joint. The above differs from [10], which also conditions on the recipient and risks bias. Following [15], we marginalize using the Schur complement:

$$\boldsymbol{\eta}_{M\alpha} = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\boldsymbol{\eta}_\beta, \tag{14}$$

$$\Lambda_{M\alpha} = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\beta\alpha}. \tag{15}$$

The outgoing message $m_{f_j \to x_r} \sim \mathcal{N}(\boldsymbol{\mu}_{f_j \to x_r}, \boldsymbol{\Lambda}_{f_j \to x_r}^{-1})$ is then obtained by projecting the marginal $\mathcal{N}^{-1}(\boldsymbol{\eta}_{M\alpha}, \boldsymbol{\Lambda}_{M\alpha})$ back to the manifold. While each factor minimizes its own energy, the GBP process aims at minimizing the global cost

$$\mathrm{E}_{\mathrm{tot}} = \sum_{j \in \mathcal{J}} \frac{1}{2} \|\mathbf{e}_j\|^2_{\boldsymbol{\Lambda}_j} + \sum_{i \in \mathcal{P}} \frac{1}{2} \|\mathbf{e}_i\|^2_{\mathbf{Q}_i^{-1}}, \tag{16}$$

where the first sum covers measurement factors (e.g., priors, odometry, camera) and the second the GP prior terms Eq. (9), introduced for continuous-time estimation.

3. Experiments

Our method achieves accurate continuous-time SLAM with robust trajectory estimation and efficiency comparable to state-of-the-art solvers. Unlike spline-based approaches requiring manual tuning of polynomial degrees and knot placement, our GP-based formulation relies on a single hyperparameter $\mathbf{Q}_C$, automatically inferred from data and naturally adapting to sensor noise and uncertainties [2, 3]. The prior covariance also integrates with GBP posteriors, enabling continuous uncertainty queries after optimization [4]

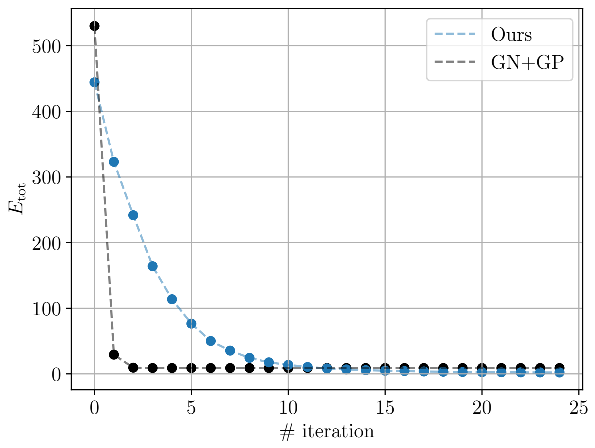


Figure 4. **Total Energy over Iterations.** G-solver and GN+GP total energy (Eq. (16)) while solving PGO with noise $\sigma = 0.1$. A G-solver iteration ends when every node is updated and has exchanged messages with all its neighbors.

Sphere		Factor noise σ					
		1.00e-04	1.00e-03	1.00e-02	1.00e-01	1.00e+00	1.50e+00
Hyperion	ATE [m]	**1.69e-04**	**1.69e-03**	1.69e-02	1.69e-01	1.69e+00	2.53e+00
Ours	ATE [m]	**1.69e-04**	**1.69e-03**	**1.68e-02**	**1.30e-01**	**7.65e-01**	**1.16e+00**

Table 1. **Interpolation Results.** Comparison of the sphere trajectory estimate queried at 100 times more frequently than control points.

(Fig. 3). We compare against Hyperion [10] and Ceres [1], both using fourth-order splines for consistency with our constant-velocity model [12]. All experiments ran on an Intel(R) Core(TM) i9-14900KF with 16 physical cores.

Evaluation and Datasets Performance is measured by Absolute Trajectory Error (ATE), the RMSE of position differences, and Absolute Rotation Error (ARE), the RMSE of orientation discrepancies computed from quaternion inner products. We tested on helix, sphere, and torus trajectories that excite all DoFs in 3D, with anisotropic noise ranging from 1e-4 to 1.5 [m] and from 1e-5 to 0.15 [rad]. We also validate on real data with a ChArUco smartphone sequence, demonstrating robustness and deployability in incremental SLAM.

Results We evaluate the solvers on prior measurements (Tab. 2), Pose Graph Optimization (PGO) (Fig. 4), incremental SLAM (Fig. 7), and dense post-optimization trajectory queries (Tab. 1), highlighting the continuous-time aspect of our approach. Synthetic tests perturb control point initialization and factors with anisotropic noise from 1e-4

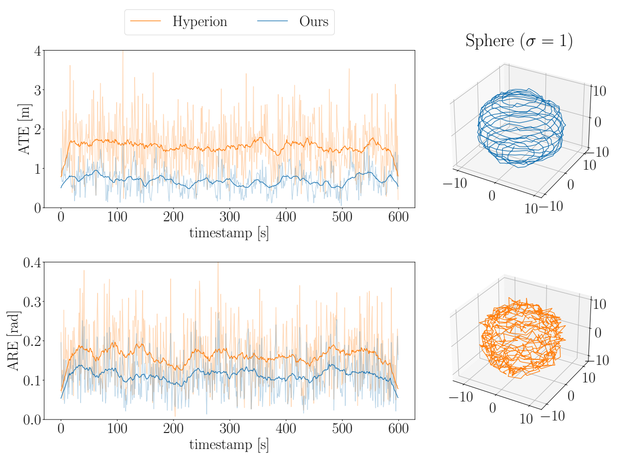


Figure 5. **Metrics along the Sphere Trajectory and Qualitative Results.** The noise levels is $\sigma = 1$. The left plots show ATE and ARE over time, while the right plots compare qualitative results for G-solver and Hyperion [10].

to 1.5 [m] and from 1e-5 to 0.15 [rad]. Since our model estimates body twists, initialization is obtained by differentiating the initial pose; for Hyperion [10] and Ceres [1], splines are sampled from the same trajectory. In Tab. 2,

Sphere		Factor noise σ					
		1.00e-04	1.00e-03	1.00e-02	1.00e-01	1.00e+00	1.50e+00
Hyperion	ATE [m]	**1.69e-04**	**1.69e-03**	1.69e-02	1.69e-01	1.69e+00	2.53e+00
	ARE [rad]	**1.78e-05**	**1.77e-04**	**1.77e-03**	1.77e-02	1.77e-01	2.66e-01
Ours	ATE [m]	**1.69e-04**	**1.69e-03**	**1.68e-02**	**1.30e-01**	**7.65e-01**	**1.16e+00**
	ARE [rad]	**1.78e-05**	**1.77e-04**	**1.77e-03**	**1.75e-02**	**1.24e-01**	**1.70e-01**

Table 2. **Prior Factors Problem.** Comparison of the sphere trajectory under varying noise levels in the measurements.

the solvers perform similarly under low noise. As noise increases, however, G-solver clearly outperforms the baselines, sometimes by an order of magnitude, as the GP prior smooths the trajectory and mitigates noise. Fig. 5 reports ATE/ARE trends and qualitative differences with Hyperion at high noise. The same results pattern arose for helix and torus trajectories. Runtime comparisons (Fig. 6) show that the continuous-time GP representation enables efficient queries, supporting online use as argued in [12]. Comparing G-solver to Gauss-Newton (GN)+GP (Fig. 4), both minimizing Eq. (3), reveals that GN converges faster due to second-order updates exploiting Hessian structure, especially in well-conditioned problems. In contrast, GBP relies on iterative message passing (synchronous in all experiments), which slows convergence in loopy graphs (e.g., PGO) due to oscillatory redundant messages. For prior factors alone, however, GBP converges more smoothly since

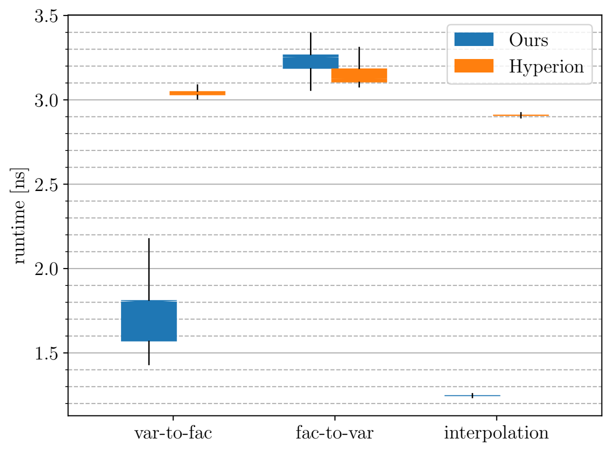


Figure 6. **Runtimes.** To compare runtimes with Hyperion [10], we exploit the factor-to-variable message, variable-to-factor message, and interpolation policy. While the first two are the core complexity of GBP methods, the last represents continuous-time estimation. The results are shown in log scale for ease of view.

the graph contains fewer loops. This highlights how graph topology directly impacts message-passing efficiency. Finally, to demonstrate real-world deployability, the experiment in Fig. 7 compares the UcoSLAM [14] pipeline (based on g2o) to our method, resulting in an ATE $= 5.8 \times 10^{-3}$ [m] and ARE $= 6.6 \times 10^{-5}$ [rad].

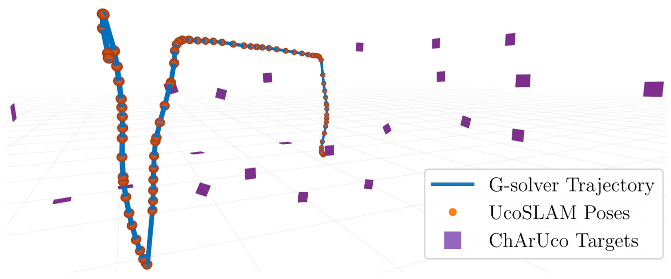


Figure 7. **ChArUco Qualitative Results.** Comparison of G-solver employed for real-world incremental SLAM and the UcoSLAM pipeline. The scene represents a smartphone camera moving in a room littered with ChArUco targets.

4. Conclusion

We presented G-solver, a continuous-time SLAM framework combining Gaussian Process priors with Gaussian Belief Propagation. By uniting smooth motion modeling with scalable inference, our method handles asynchronous sensors and uncertainty while matching state-of-the-art efficiency and improving accuracy. Future work will explore more compact message-passing schemes to reduce cost and extend the model to constant acceleration and noisy jerk for richer motion dynamics.

References

[1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres solver, 2023. 1, 3, 4

[2] Sean Anderson and Timothy D. Barfoot. Full steam ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on se(3). In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, page 157–164. IEEE Press, 2015. 2, 3

[3] Sean Anderson, Timothy D Barfoot, Chi Hay Tong, and Simo Särkkä. Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression. *Autonomous Robots*, 39:221–238, 2015. 3

[4] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2024. 3

[5] Tim D. Barfoot, Chi Hay Tong, and Simo Särkkä. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Proc. of Robotics: Science and Systems (RSS)*, 2014. 1, 2

[6] Danny Bickson. Gaussian belief propagation: Theory and aplication. *CoRR*, abs/0811.2518, 2008. 1

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. 2

[8] Paul Furgale, Timothy Barfoot, and Gabe Sibley. Continuous-time batch estimation using temporal basis functions. *Int. J. Robotic Res.*, 34:2088–2095, 2012. 1

[9] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77, 2013. 1

[10] David Hug, Ignacio Alzugaray, and Margarita Chli. Hyperion – a fast, versatile symbolic gaussian belief propagation framework for continuous-time slam. In *ECCV*, page 215–231, Berlin, Heidelberg, 2024. Springer-Verlag. 1, 2, 3, 4

[11] Jiarong Jiang, Piyush Rai, and Hal Daume. Message-passing for approximate map inference with latent variables. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2011. 2

[12] Jacob Johnson, Joshua Mangelson, Timothy Barfoot, and Randal Beard. Continuous-time trajectory estimation: A comparative study between gaussian process and spline-based approaches, 2024. 3, 4

[13] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics*, 24(6):1365–1378, 2008. 1

[14] Rafael Munoz-Salinas and Rafael Medina-Carnicer. Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers. *Pattern Recognition*, 101:107193, 2020. 4

[15] J. Ortiz. *Gaussian Belief Propagation for Real-time Decentralised Inference*. Imperial College London, 2023. 1, 2, 3

[16] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. 1