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

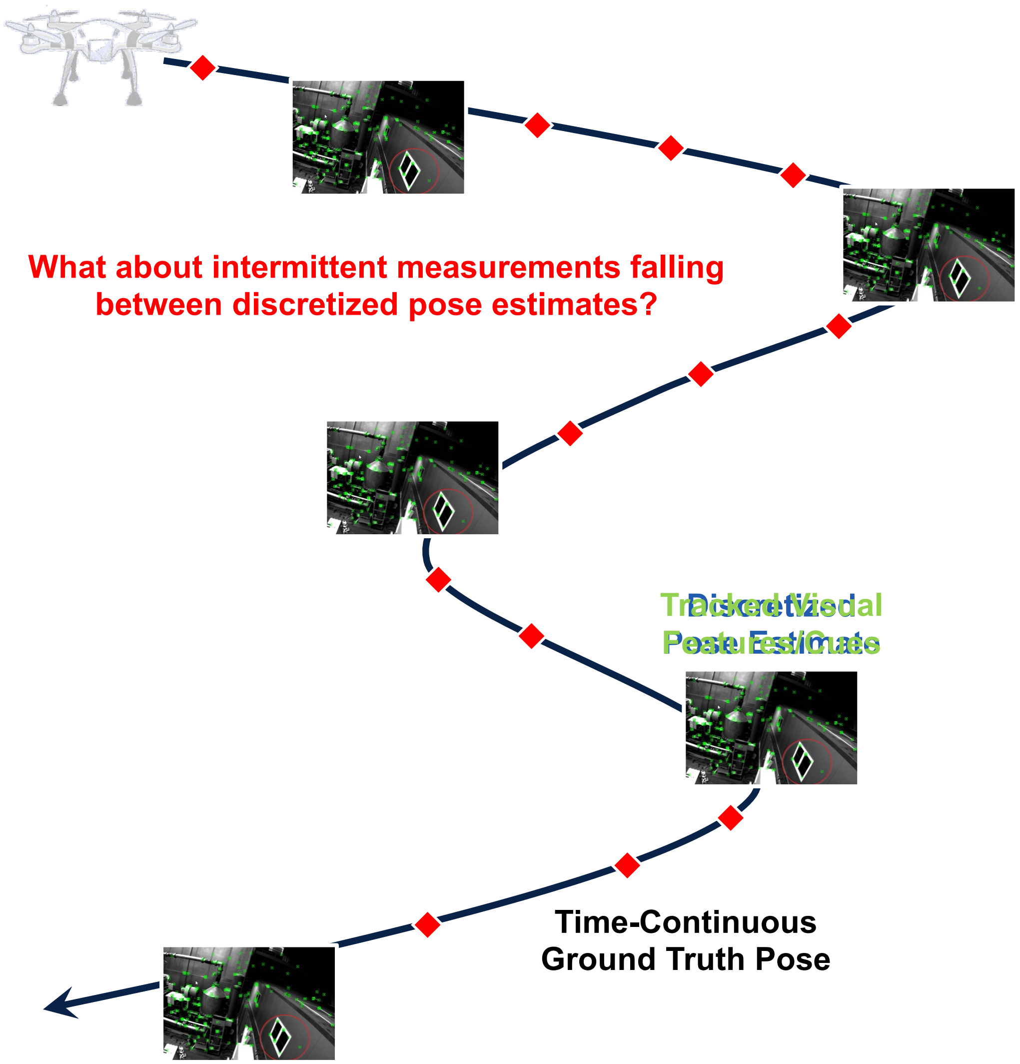
David Hug[1], Ignacio Alzugaray[2], Margarita Chli[1]

[1]Vision for Robotics Lab (V4RL), ETH Zurich and University of Cyprus
[2]Dyson Robotics Lab, Imperial College London

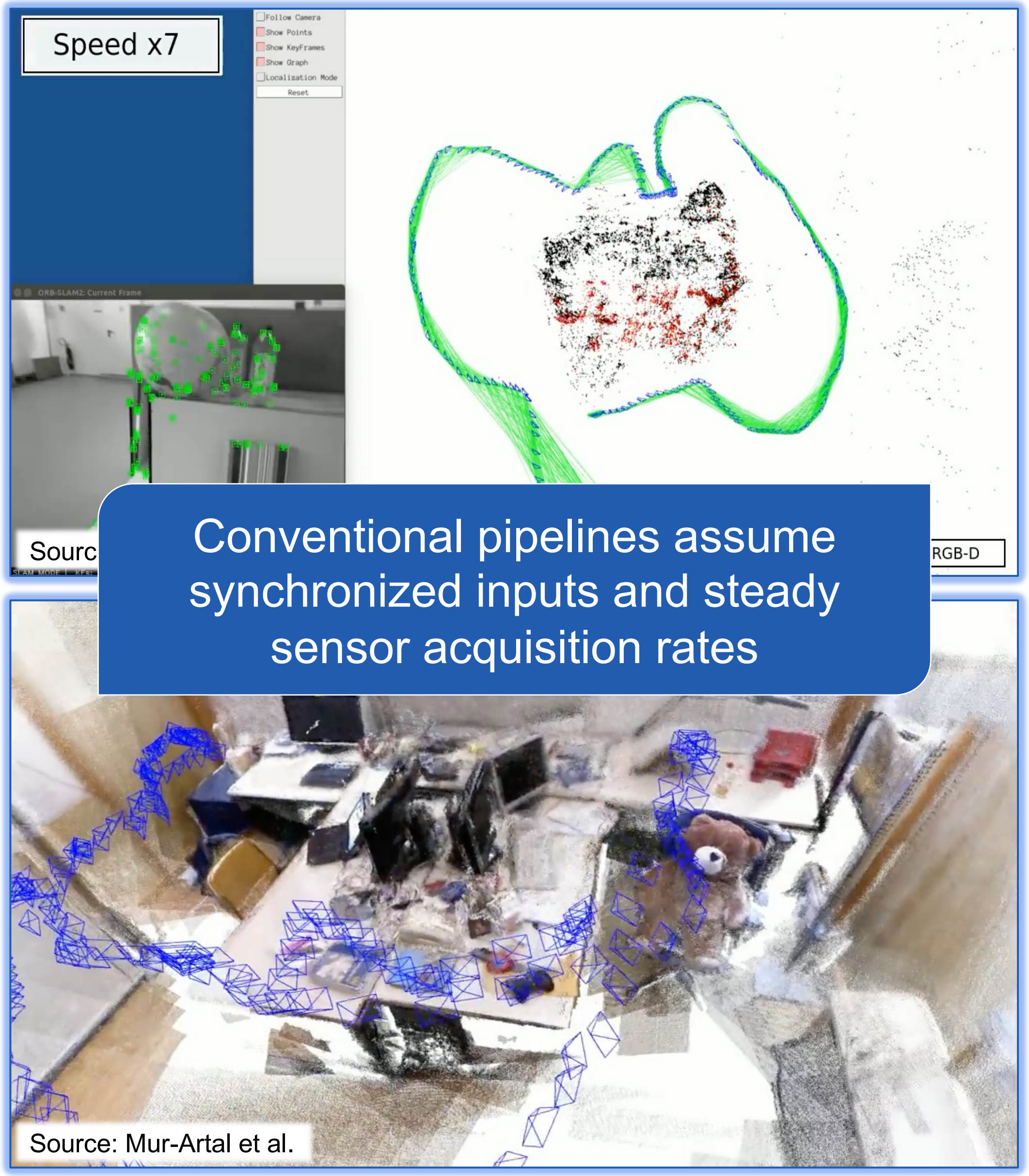
Presented at the European Conference on Computer Vision 2024

Related Work

Discrete-Time SLAM



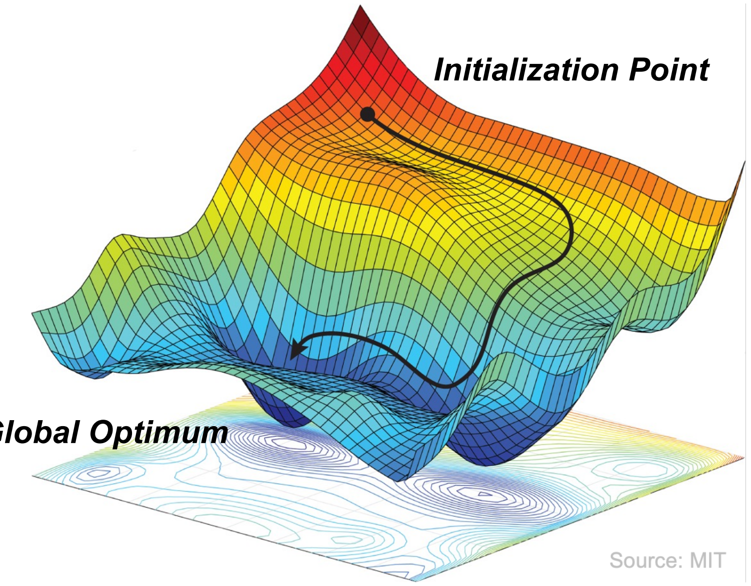
What about intermittent measurements falling between discretized pose estimates?



Conventional pipelines assume synchronized inputs and steady sensor acquisition rates

Related Work

Non-Linear Least Squares Optimization (Single Agent)



Surface Plot of Cumulative Residuals

***Minimization Problem*[1]**

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} \|\bar{\mathbf{r}}(t, \theta)\|^2 \right] \text{ with } \theta \subseteq \boldsymbol{\Theta}$$

Optimal Parameters — *Sum over square of weighted residuals stemming from all sensors and measurement times*

Weighted Residual

$$\|\bar{\mathbf{r}}(t, \theta)\|^2 = \bar{\mathbf{r}}^\top \bar{\mathbf{r}} = \mathbf{r}^\top \Sigma_m^{-1} \mathbf{r}$$

Precision

Residual

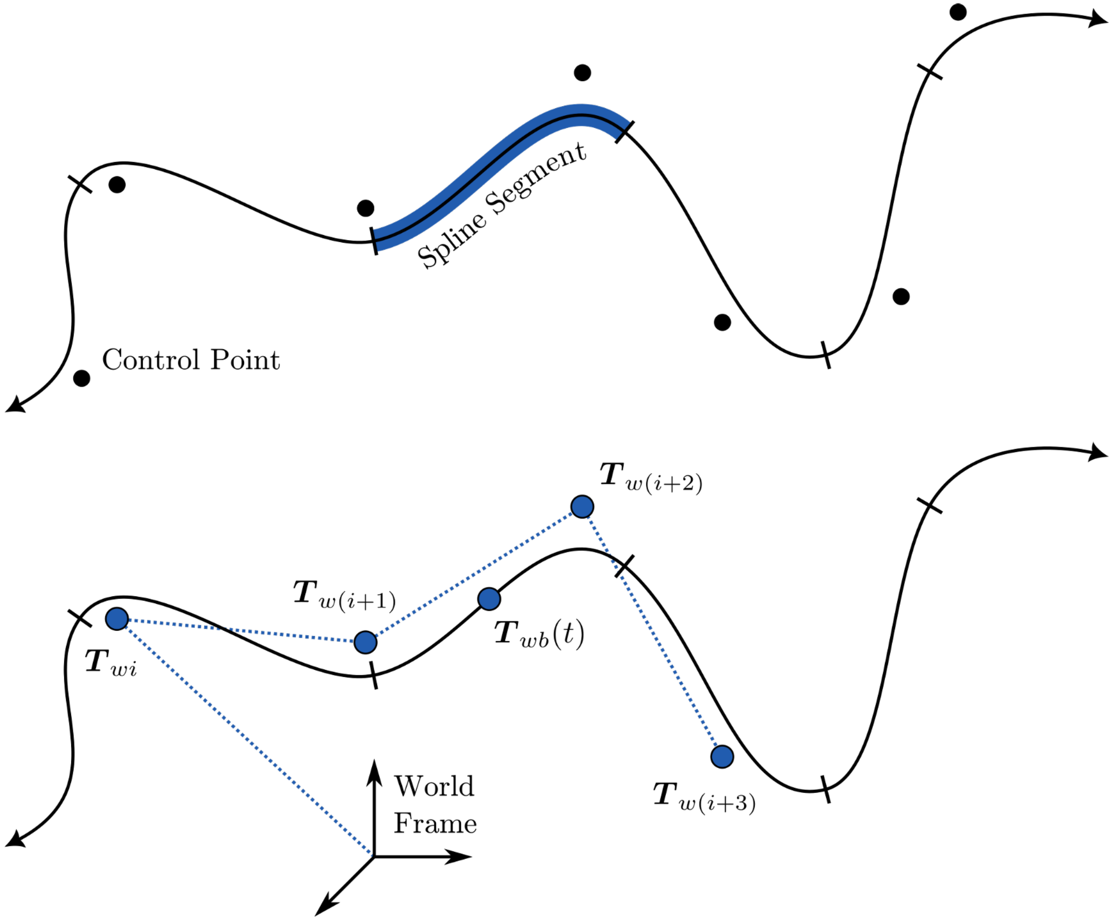
Distance Metric

$$\mathbf{r}(t, \theta) = \hat{\mathbf{m}}(t, \theta) \boxminus_\mu \mathbf{m}(t, \theta)$$

Predicted Measurement — *Actual Measurement*

Related Work

Continuous-Time SLAM[2]



Spline Degree ***Interpolation Coefficient***

$$\boldsymbol{T}_{wb}(t) = \underbrace{\boldsymbol{T}_{wi}}_{\text{Base Control Point}} \prod_{j=1}^{k} \underbrace{\left(\boldsymbol{T}_{w(i+j-1)}^{-1}\,\boldsymbol{T}_{w(i+j)}\right)^{\lambda_{i,j}(t)}}_{\text{Incremental Elements in the Lie Group}} \in \mathbb{SE}(3)$$

$$= \underbrace{\exp\left(\lambda_j(t)\log\left(\boldsymbol{T}_{w(i+j-1)}^{-1}\,\boldsymbol{T}_{w(i+j)}\right)\right)}_{\text{Geodesic Curves}}$$

$$\Lambda_i(t) = \begin{bmatrix}\lambda_{i,0}(t)\\ \lambda_{i,1}(t)\\ \lambda_{i,2}(t)\\ \lambda_{i,3}(t)\end{bmatrix} = \frac{1}{3!}\begin{bmatrix}6 & 0 & 0 & 0\\ 5 & 3 & -3 & 1\\ 1 & 3 & 3 & -2\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u^0(t)\\ u^1(t)\\ u^2(t)\\ u^3(t)\end{bmatrix}$$

B-Spline Interpolation Coefficients[1] ***Vector of Normalized Times***

[1] Qin, PCCGA 1998
[2] Hug et al., RA-L 2022

Methodology
Approach

Mathematical expressions for spline-related ***residuals remain tedious and error-prone***, leading to ***suboptimal performance***

Standard NLLS optimizers ***neither model uncertainties*** nor ***do they (trivially) extend to distributed computations*** across multiple agents

Approach
Leverage Gaussian Belief Propagation (GBP) for ***distributed, stochastic inference*** along with ***automating code generation***

Methodology
Code Generation

Approach
Extend SymForce[1] to ***delegate the generation of performance-critical code*** within the framework

High-level, symbolic mathematical expressions and residuals

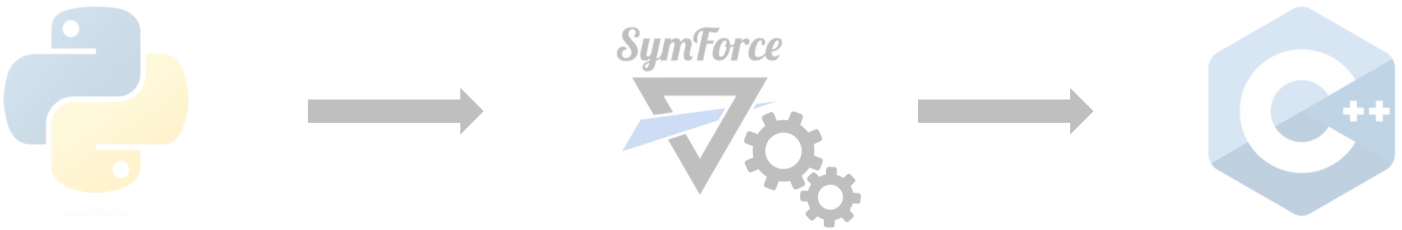
Translation of complex expressions and symbolic optimization

High-performance machine code lacking interpretability

[1] Martiros et al., RSS 2022

Experiments

Code Generation



High-level, symbolic mathematical expressions and residuals

Translation of complex expressions and symbolic optimization

High-performance machine code lacking interpretability

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

Performance comparison between our auto-generated and optimized B-Spline implementation and the hand-crafted implementation from Sommer et al. [1]

[1] Sommer et al., CVPR 2020

Experiments

Code Generation

High-level, symbolic mathematical expressions and residuals

Translation of complex expressions and symbolic optimization

High-performance machine code lacking interpretability

Setup			Pose [s]		Velocity [s]		Acceleration [s]		Avg. Speedup
$\mathcal{L}$	k	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
$\mathbb{SO}(3)$	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2.64x
$\mathbb{SO}(3)$	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	17.05x
$\mathbb{SO}(3)$	5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2.58x
$\mathbb{SO}(3)$	5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	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

Insight
Residual and Jacobian evaluations represent ***a major performance-limitation***, rendering 110x speedups momentous

Performance comparison between our auto-generated and optimized B-Spline implementation and the hand-crafted implementation from Sommer et al. [1]

[1] Sommer et al., CVPR 2020

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

Minimization Problem[1]

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} \|\bar{\mathbf{r}}(t, \theta)\|^2 \right] \text{ with } \theta \subseteq \boldsymbol{\Theta}$$

[1] Agarwal et al., Ceres Solver

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

Minimization Problem[1]

$$\mathbf{\Theta}^* = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} \|\bar{\mathbf{r}}(t, \theta)\|^2 \right] \text{ with } \theta \subseteq \mathbf{\Theta}$$

Simplification

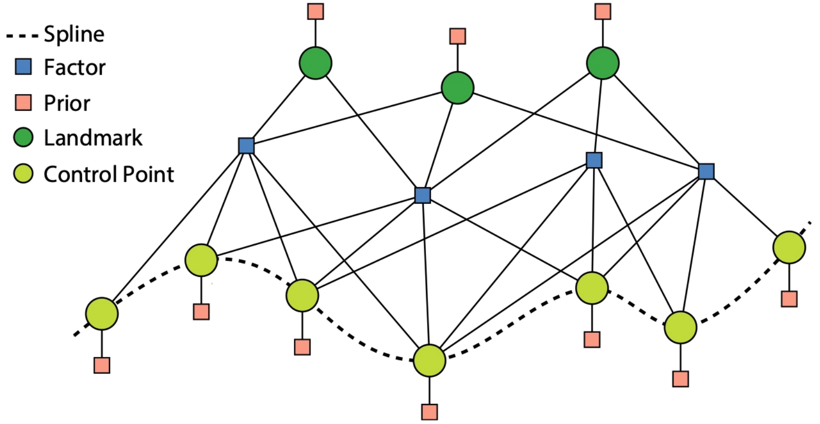
Stochastic Optimization Problem[2,3]

$$\mathbf{\Theta}^* = \underset{\mathbf{\Theta}}{\operatorname{argmax}} \log\left(p\left(\mathbf{\Theta}\right)\right) = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \sum_{i} \|\bar{\mathbf{r}}_i\left(t_i, \theta_i\right)\|^2$$

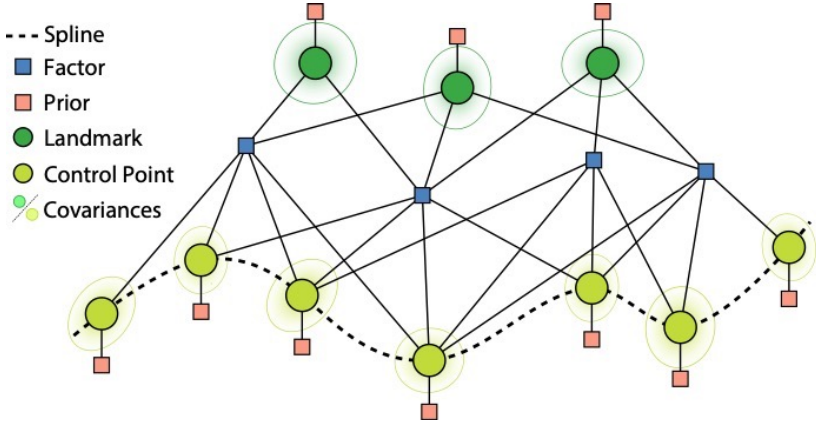
Probability Distribution ***Residuals***

$$p\left(\mathbf{\Theta}\right) = \prod_i f_i\left(t_i, \theta_i\right) \propto \prod_i \exp\left(-E_i\left(t_i, \theta_i\right)\right)$$

Assume Gaussian Distributions



Continuous-Time Factor Graph (Visual)



Stochastic Continuous-Time Factor Graph (Visual)

[1] Agarwal et al., Ceres Solver
[2] Ortiz et al., CVPR 2020
[3] Ortiz et al., ICRA 2022

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

Minimization Problem[1]

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} \| \bar{\mathbf{r}}(t, \theta) \|^2 \right] \text{ with } \theta \subseteq \Theta$$

Simplification

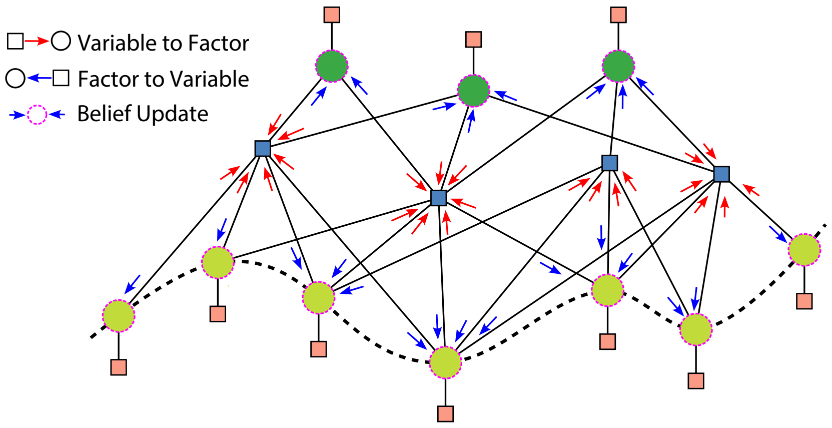
Stochastic Optimization Problem[2,3]

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \log \left(p \left(\Theta \right) \right) = \underset{\Theta}{\operatorname{argmin}} \sum_i \| \bar{\mathbf{r}}_i \left(t_i, \theta_i \right) \|^2$$

Probability Distribution ***Residuals***

$$p \left(\Theta \right) = \prod_i f_i \left(t_i, \theta_i \right) \propto \prod_i \exp \left(-E_i \left(t_i, \theta_i \right) \right)$$

Assume Gaussian Distributions



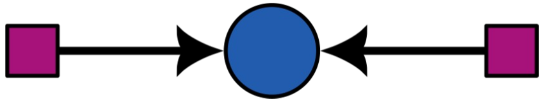
Local Message Passing in the Factor Graph

[1] Agarwal et al., Ceres Solver
[2] Ortiz et al., CVPR 2020
[3] Ortiz et al., ICRA 2022

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

Node Update



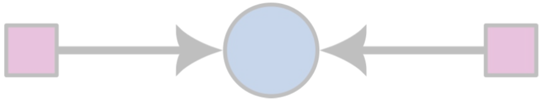
$$\eta_{n_j} = \sum_{f_i \in N(n_j)} \eta_{f_i \to n_j}$$

"What do my neighbors believe about me?"

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

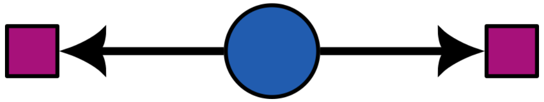
Node Update



$$\eta_{n_j} = \sum_{f_i \in N(n_j)} \eta_{f_i \to n_j}$$

"What do my neighbors believe about me?"

Node-to-Factor Messages

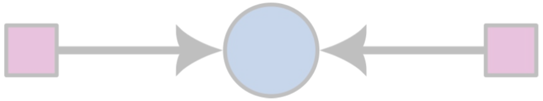


"Pass the updated belief to neighboring factors"

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

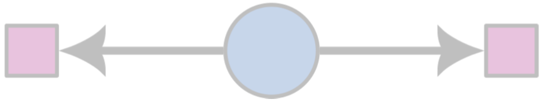
Node Update



$$\eta_{n_j} = \sum_{f_i \in N(n_j)} \eta_{f_i \rightarrow n_j}$$

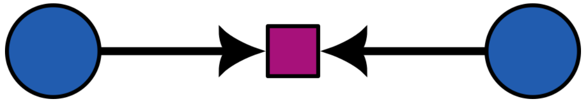
"What do my neighbors believe about me?"

Node-to-Factor Messages



"Pass the updated belief to neighboring factors"

Factor Update



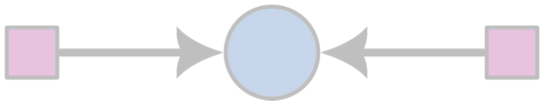
$$\eta_i^0 = -\bar{\mathbf{J}}_i^{0,\top} \bar{\mathbf{r}}_i^0$$

"Reevaluate the Residual and Jacobian"

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

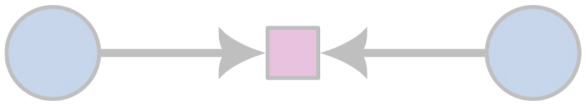
Node Update



$$\eta_{n_j} = \sum_{f_i \in N(n_j)} \eta_{f_i \to n_j}$$

"What do my neighbors believe about me?"

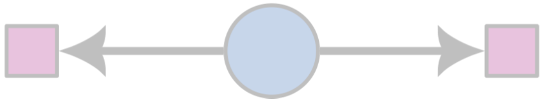
Factor Update



$$\eta_i^0 = -\bar{\mathbf{J}}_i^{0,\top} \bar{\mathbf{r}}_i^0$$

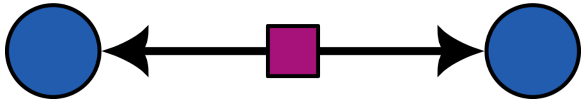
"Reevaluate the Residual and Jacobian"

Node-to-Factor Messages



"Pass the updated belief to neighboring factors"

Factor-to-Node Messages



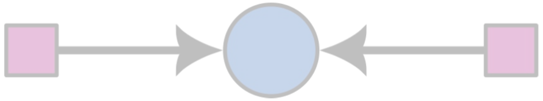
$$\eta'_{f_i \to n_a} = \eta_a^0 - \mathbf{\Lambda}'^{\top}_{aa} \mathbf{\Lambda}'^{-1}_{bb} \eta'_b$$

"Marginalize the probability distribution for a neighboring node"

Methodology

Continuous-Time Gaussian Belief Propagation (GBP)

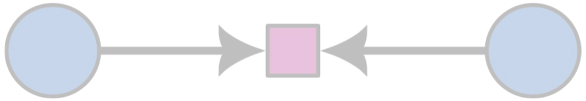
Node Update



$$\eta_{n_j} = \sum \eta_{f_i \to n_j}$$

“What do n

Factor Update



$$\eta_i^0 = -\mathbf{J}_i^{0,\top}\mathbf{r}_i^0$$

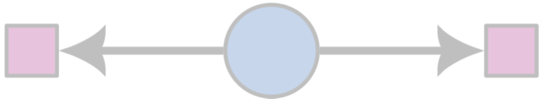
Jacobian”

Approach

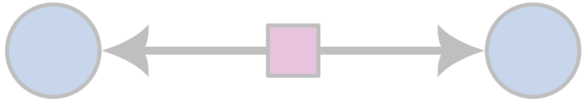
Evaluate the suitability of the ***combined GBP-based CT-SLAM method*** in absolute and localization setups

Node-to

s



“Pass the updated belief to neighboring factors”



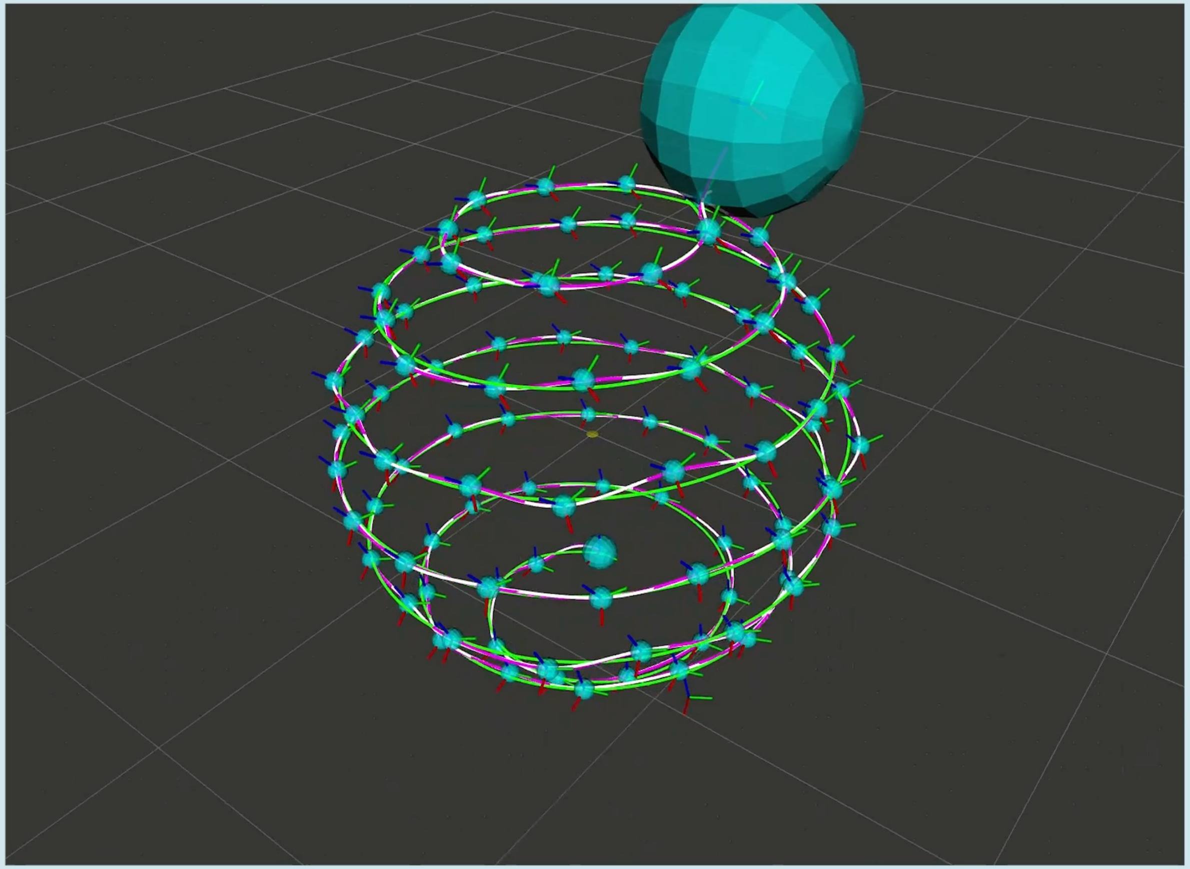
$$\eta'_{f_i \to n_a} = \eta_a^0 - \Lambda'^{\top}_{aa}\Lambda'^{-1}_{bb}\eta'_b$$

“Marginalize the probability distribution for a neighboring node”

Experiments

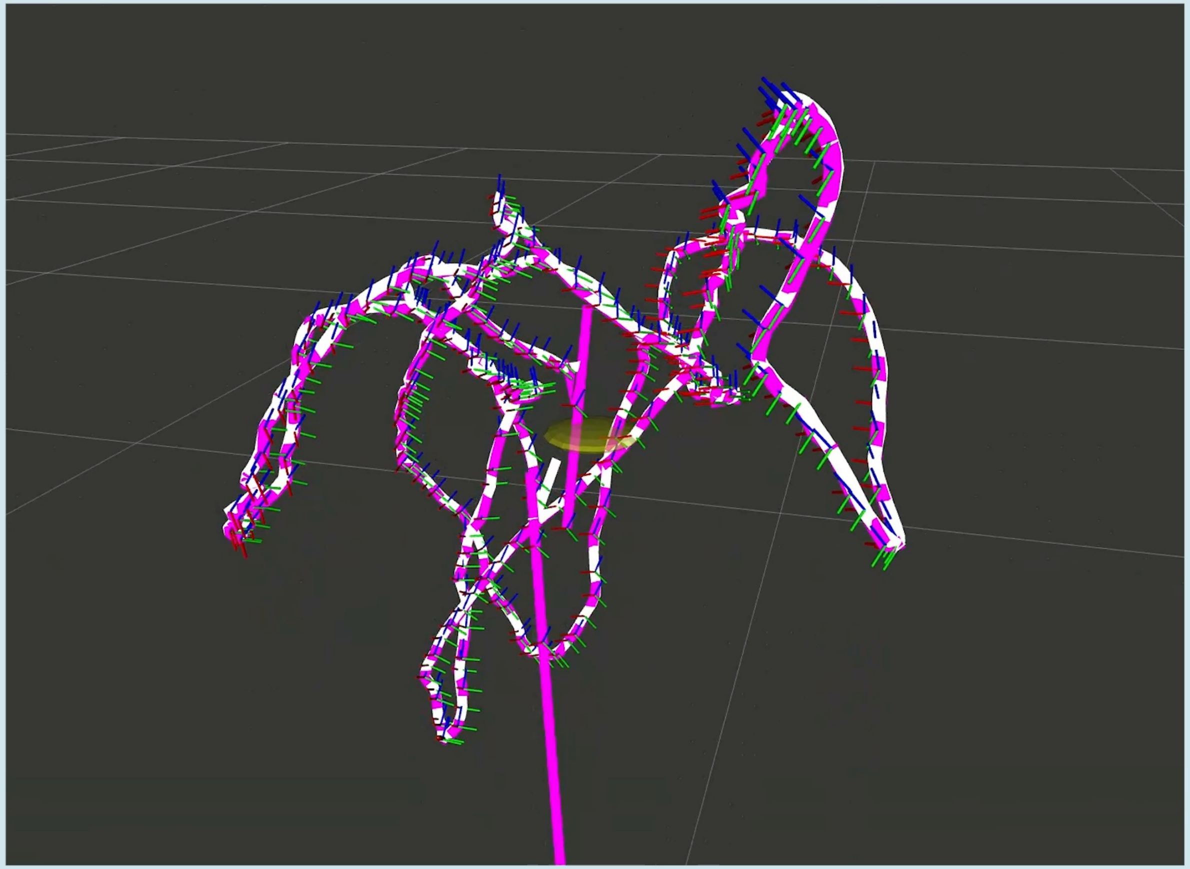
Motion Capture Setups

Absolute Setup (Simulation)



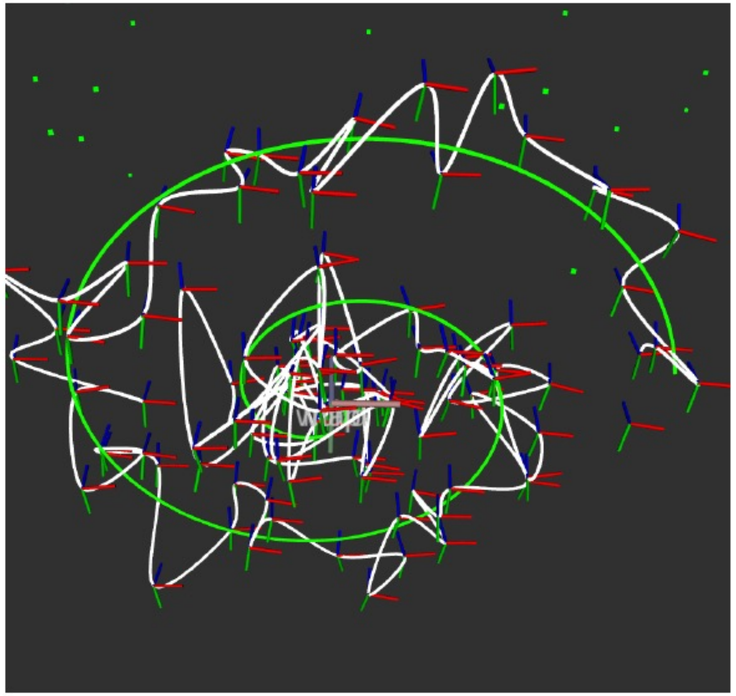
In slow motion: The proposed continuous-time GBP solver (in magenta) and the conventional Ceres solver (in white) converge to identical solutions close to the ground truth (in green) even under poor initialization (±1.00 m/rad) and substantial pose measurement noise (±0.05 m/rad).

Absolute Setup (ChArUco)

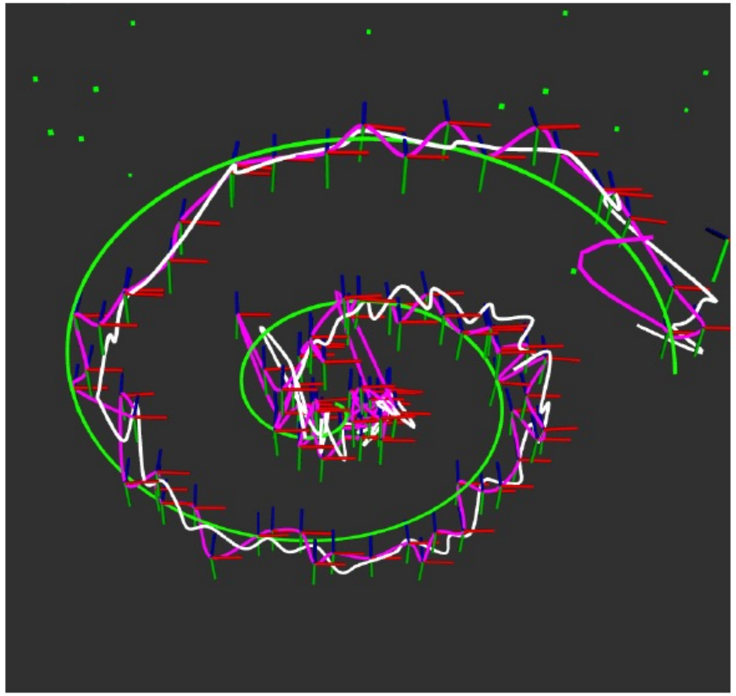


In slow motion: The proposed GBP solver (in magenta) and Ceres (in white) converge to identical solutions close in the ChArUco experiment (with initialization at identity).

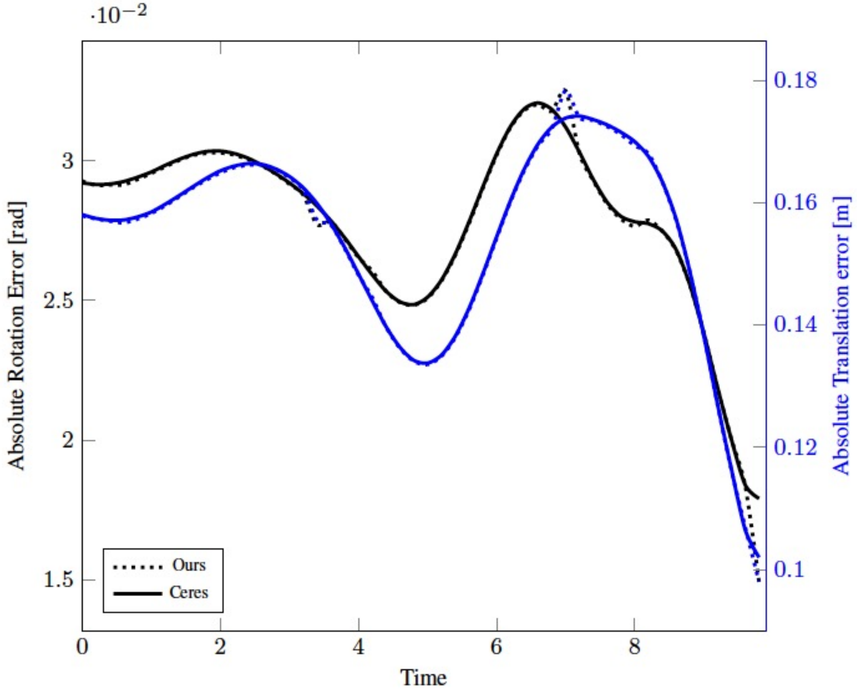
Experiments
Localization Setup



(a) At initialization



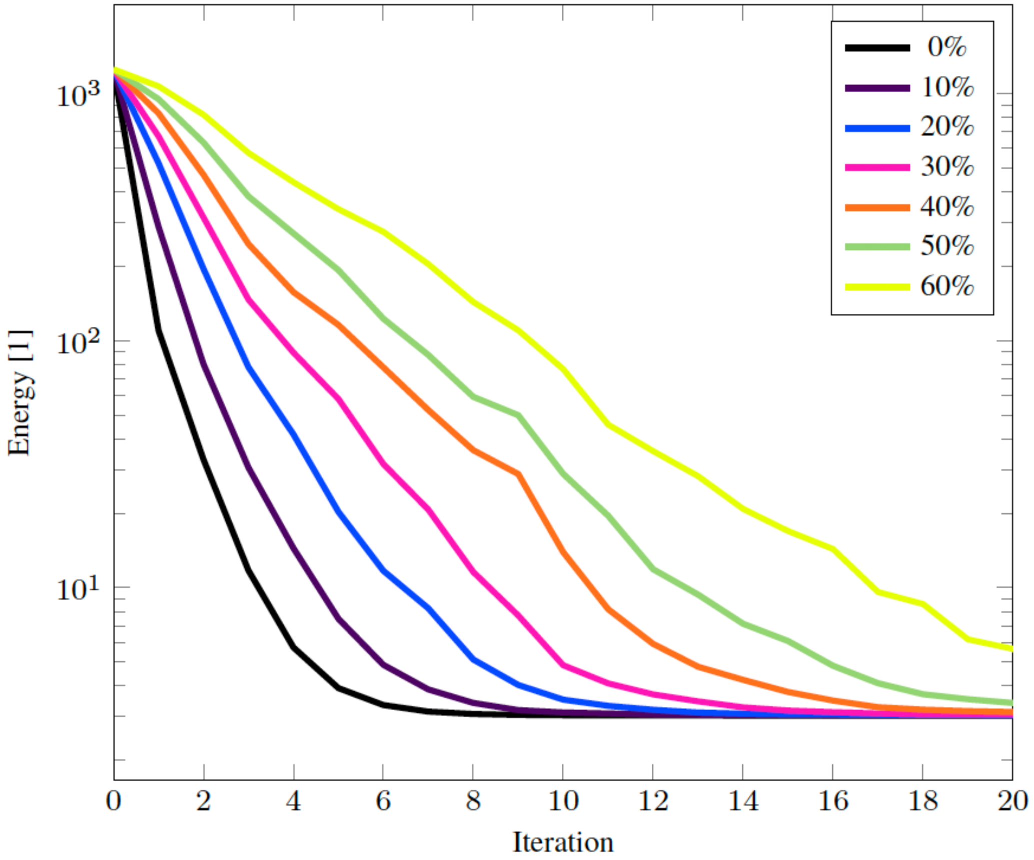
(b) After the 1[st] iteration



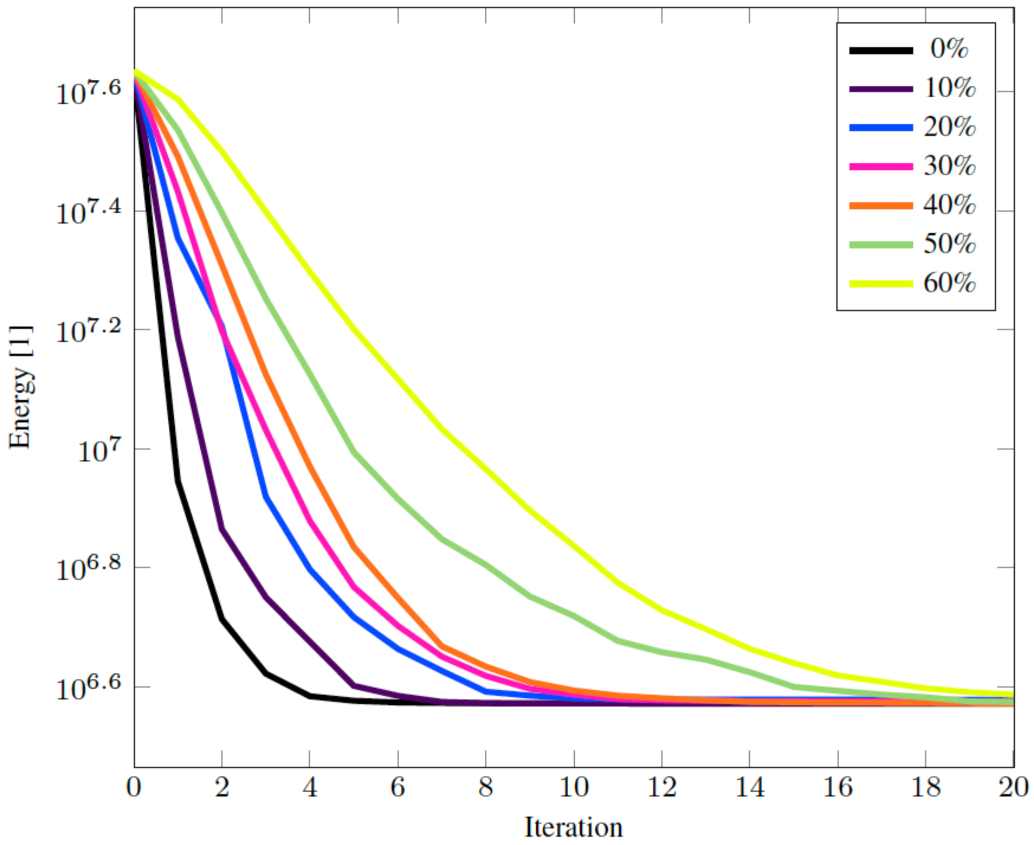
(c) Absolute Error

Left: Estimated motion yielded by the proposed GBP-based framework (in magenta) and Ceres (in white) across iterations and plotted against ground truth (in white). Right: Resulting errors from Hyperion and Ceres are identical.

Experiments
Ablation on Message Dropouts



(a) Dropout Convergence: Absolute

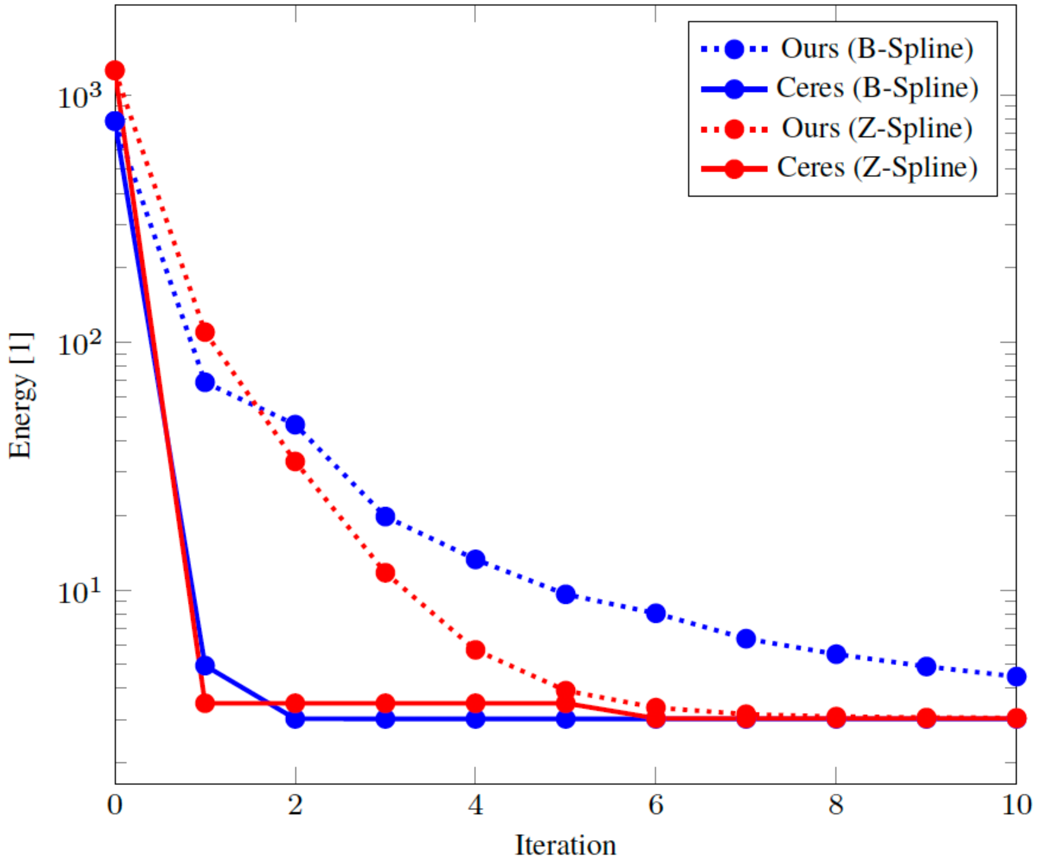


(b) Dropout Convergence: Localization

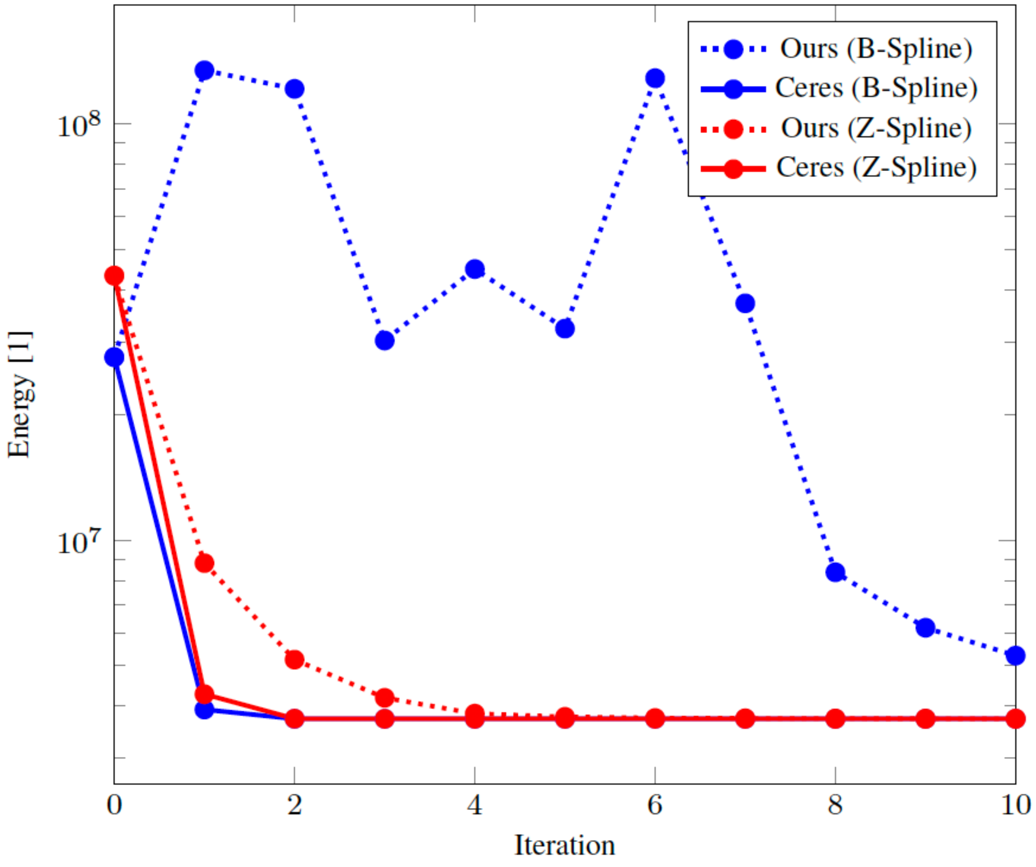
Graph energy vs. number of iterations conditioned on the message dropout percentage.

Experiments

Ablation on B- and Z-Splines



(a) Convergence: Absolute



(b) Convergence: Localization

Graph energy conditioned on the spline and solver type across iterations.

Conclusions

1.) Presents the ***first open-source GBP-based continuous-time optimization framework*** with ***symbolic code generation***

Conclusions

1.) Presents the ***first open-source GBP-based continuous-time optimization framework*** with ***symbolic code generation***

2.) Implements the ***fastest, Ceres-interoperable[1] B- and Z-Spline implementations*** to date, further alleviating computational limitations

[1] Agarwal et al., Ceres Solver

Conclusions

1.) Presents the ***first open-source GBP-based continuous-time optimization framework*** with ***symbolic code generation***

2.) Implements the ***fastest, Ceres-interoperable[1] B- and Z-Spline implementations*** to date, further alleviating computational limitations

3.) ***Demonstrates the efficacy*** of the proposed framework ***in absolute and localization setups***

[1] Agarwal et al., Ceres Solver

Find us on GitHub!