Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

VISION FOR ROBOTICS LAB
Vision for Robotics Lab
Prof. Margarita Chli

Semester Thesis

Gaussian Belief Propagation for Continuous Time Simultaneous Localization and Mapping

Spring Term 2024

Supervised by:
David Hug
Cornelius von Einem

Author:
Hojune Kim

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Gaussian Belief Propagation for Continuous Time Simultaneous Localization and Mapping

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Hojune	Kim

Student supervisor(s)

David	Hug
Cornelius	Einem

Supervising lecturer

Margarita	Chli

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zürich, 14.08.24

Place and date

Hojune Kim [illegible]

Signature

[1]Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

Conventional Simultaneous Localization and Mapping (SLAM) methods, typically based on discrete-time formulations, have demonstrated significant utility in various robotics applications. These methods, however, face challenges such as quantization errors and inefficiencies in handling asynchronous data from multiple sensors. Continuous-time SLAM approaches, utilizing splines to represent motion, offer a promising alternative by more naturally integrating sensor data captured at irregular intervals. Despite their potential, many continuous-time SLAM methods rely on centralized Non-Linear Least Squares (NLLS) solvers, which can be computationally expensive and inefficient for asynchronous data processing.

This project addresses challenges in SLAM by integrating the decentralized solver Gaussian Belief Propagation (GBP) with a continuous-time formulation, specifically utilizing Z-splines. The approach aims to enhance robustness and efficiency in multi-robot systems. The method is validated using a real-world dataset and compared against traditional NLLS solvers. Results demonstrate that GBP with continuous-time representation outperforms NLLS in handling perturbations, providing more stable and efficient optimization.

Symbols

Symbols

$\bar{\boldsymbol{r}}$	weighted residuals
$\boldsymbol{\mu}$	mean
$\boldsymbol{\Sigma}$	covariance matrix
$\boldsymbol{\eta}$	information vector
$\boldsymbol{\Lambda}$	precision matrix
$\boldsymbol{\theta}_i$	optimization parameters
$\boldsymbol{m}_{f_i \to n_j}$	factor-to-node message
$\mathcal{N}^{-1}$	Gaussian distribution in canonical form
$\mathfrak{N}$	Mixed Gaussian Representation for Lie groups
α_{n_j}	step size for node update
α_{f_i}	step size for factor update

Acronyms and Abbreviations

SLAM	Simultaneous Localization and Mapping
GBP	Gaussian Belief Propagation
NLLS	Non-Linear Least Squares
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
ADMM	Alternating Direction Method of Multipliers
DoF	Degree of Freedom
RMSE	Root Mean Square Error
PnP	Perspective-n-Point
BA	Bundle Adjustment

Chapter 1

Introduction

In the realm of robotics, Simultaneous Localization and Mapping (SLAM) is crucial for enabling robots to generate a map of an unknown environment while simultaneously tracking their pose within it. Conventional SLAM approaches have predominantly utilized discrete-time formulations and have leveraged a range of sensors, including visual cameras [1], IMUs [2], and LiDAR [3]. While these discrete methods have demonstrated strengths in various applications, they also present notable limitations.

Traditionally, SLAM optimizes a discrete set of key frames to manage the non-continuous acquisition of sensor data and to control the growth of the optimization problem. This approach has made discrete formulations popular for many robotics applications due to their effectiveness in handling sensor data. However, discrete optimization does not account for inter-state motion, leading to quantization errors when integrating or synchronizing asynchronous data from multiple sensors. For example, IMU data often requires pre-integration to align with other constraints [4], and LiDAR scan lines or rolling-shutter cameras face challenges with timing discrepancies between measurements.

To address these issues, continuous-time approaches that use splines to represent motion have been proposed [3, 5]. These methods can effectively utilize asynchronous data to optimize spline states. However, many of these approaches rely on conventional Non-Linear Least Squares (NLLS) solvers, which are centralized and require re-solving the problem each time new measurements are obtained from asynchronous sensors.

These limitations become increasingly problematic in multi-robot systems, where both asynchronous data from intermittent robot communication and the centralized nature of traditional SLAM solvers create bottlenecks. To overcome these challenges and improve real-time performance and resource utilization, distributed solving methods like Gaussian Belief Propagation (GBP) have emerged [6].

Our project addresses these challenges by employing a continuous-time representation for SLAM, specifically leveraging GBP frameworks. This approach is motivated by the need to handle asynchronous sensor data effectively while maintaining robust and efficient optimization in distributed systems. Continuous-time SLAM offers a more natural and flexible framework for integrating sensor data acquired at irregular intervals, providing significant advantages over discrete-time methods, especially in distributed multi-robot systems.

Chapter 2

Related Work

SLAM (Simultaneous Localization and Mapping) has traditionally focused on discrete-time approaches, demonstrating promising performance in both monocular [2, 7] and stereo setups [1]. Additionally, discrete-time SLAM has been effectively combined with various sensing modalities, such as IMUs [8, 9] and LiDAR [3], to enhance its robustness and accuracy.
In contrast to these well-established discrete-time SLAM methods, continuous-time SLAM techniques have emerged, offering significant potential for continuous motion estimation and high-fidelity results [10, 5, 11]. Continuous-time SLAM methods are designed to seamlessly integrate unsynchronized and asynchronous measurements throughout the estimation process. Despite these advantages, many continuous-time SLAM techniques still face challenges related to computational efficiency and convergence. Addressing these issues remains a key area of ongoing research.
Beyond fundamental SLAM research focused on single agents, a new challenge has emerged in deploying these techniques in collaborative multi-agent systems. Previous work, such as [**?**], has explored methods for centralized multi-agent SLAM. Strategies for distributed NLLS optimizations often employ approaches like the Alternating Direction Method of Multipliers (ADMM), as proposed in [12] and [13]. ADMM approaches utilize dual residuals to ensure consistent estimates across distributed NLLS optimizations.
Alternatively, Gaussian Belief Propagation (GBP) methods [14, 15, 6] provide an efficient framework for distributed and asynchronous state inference through message-passing. GBP is particularly suited for multi-agent SLAM due to its ability to propagate beliefs across a network of agents. Furthermore, GBP has been applied to continuous-time formulations in works like [16], which utilize B-splines and Z-splines to represent motion. Despite these advancements, there remains a lack of comprehensive verification across various scenarios.
In this project, we build on these concepts by integrating distributed GBP optimization with continuous-time parameterization. We aim to verify the effectiveness of this combined approach in practical applications, addressing the gaps identified in current research.

Chapter 3

Methodology

3.1 Continuous Time Parameterization

Many studies have explored continuous-time representations in conventional SLAM algorithms, with cubic Z-splines being a common parameterization for motion. An analytic continuous spline can be defined by control points, where each segment of the spline is associated with a set of control points $\{B_i, \ldots, B_{i+k}\}$. Here, k denotes the Degree of Freedom (DoF) of the Z-spline, which is set to 4 in this project. Starting from the world-to-body transformation of the first control point $T_{wi}(t) \in \mathbb{SE}(3)$, the transformation of a query point can be computed as an interpolation between the motions of the control points, with coefficients found in [17].

$$\boldsymbol{T}_{wb}(t) = \begin{bmatrix} \boldsymbol{R}_{wb}(\boldsymbol{q}_{wb}(t)) & \boldsymbol{t}_{wb}(t) \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{SE}(3) \text{ with} \tag{3.1}$$

$$\boldsymbol{q}_{wb}(t) = \boldsymbol{q}_{wi} * \prod_{j=1}^{k} \left(\boldsymbol{q}_{w(i+j-1)}^{-1} * \boldsymbol{q}_{w(i+j)} \right)^{\lambda_j(t)} \tag{3.2}$$

$$\boldsymbol{t}_{wb}(t) = \boldsymbol{t}_{wi} + \sum_{j=1}^{k} \left[\lambda_j(t) \left(\boldsymbol{t}_{w(i+j)} - \boldsymbol{t}_{w(i+j-1)} \right) \right], \tag{3.3}$$

3.2 Continuous Time Optimization

Since the data received from the sensors are noisy, various SLAM algorithms focus on minimizing the cost function by optimizing parameters Θ. Most methods solve the optimal solution by minimizing the summation of the weighted residuals from the measurements. Weighted residuals r are the difference between predicted measurements based on the sensor parameters and real measurements. These residuals are weighted by the square-root information matrix $\boldsymbol{\Omega}_m$, which reflects the characteristics of the sensors to balance each residual. The conventional method, Non-linear Least Squares (NLLS) solvers, aims to obtain optimal parameters by minimizing the sum of residual costs as shown below:

$$\boldsymbol{r}(t, \boldsymbol{\theta}_s) = \hat{\boldsymbol{m}}(t, \boldsymbol{\theta}_s) \boxminus_{\boldsymbol{\mu}} \boldsymbol{m}(t) \text{ and} \tag{3.4}$$

$$\|\bar{\boldsymbol{r}}\|^2 = \bar{\boldsymbol{r}}^\top \bar{\boldsymbol{r}} = \boldsymbol{r}^\top \boldsymbol{\Omega}_m^\top \boldsymbol{\Omega}_m \boldsymbol{r} = \boldsymbol{r}^\top \boldsymbol{\Lambda}_m \boldsymbol{r} = \boldsymbol{r}^\top \boldsymbol{\Sigma}_m^{-1} \boldsymbol{r}, \tag{3.5}$$

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \left[\sum\nolimits_{s \in \mathcal{S}} \sum\nolimits_{t \in \mathcal{T}_s} \frac{1}{2} \|\bar{\boldsymbol{r}}(t, \boldsymbol{\theta}_s)\|^2 \right]. \tag{3.6}$$

where $\boldsymbol{\Sigma_m}$ and $\boldsymbol{\Lambda_m}$ are covariance and precision matrices.
By using a probabilistic formulation, the non-linear optimization problem can also be represented as a product of factors $f_i \propto e^{-E_i(\boldsymbol{\theta}_i)}$ with induced energies $E_i(\boldsymbol{\theta}_i)$.

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \log\left(p\left(\boldsymbol{\Theta}\right)\right) = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_i E_i(t_i, \boldsymbol{\theta}_i) \text{ with} \tag{3.7}$$

$$p\left(\boldsymbol{\Theta}\right) = \prod_i f_i\left(t_i, \boldsymbol{\theta}_i\right) \propto \prod_i e^{-E_i(t_i, \boldsymbol{\theta}_i)}. \tag{3.8}$$

The weighted residual $\bar{\boldsymbol{r}}_i$ can be approximated by Taylor expansions in $\boldsymbol{\theta}_i$ around a linearization point $\boldsymbol{\theta}_i^0$.

$$\bar{\boldsymbol{r}}_i(\boldsymbol{\theta}_i) - \bar{\boldsymbol{r}}_i(\boldsymbol{\theta}_i^0) \approx D\bar{\boldsymbol{r}}_i(\boldsymbol{\theta}_i^0)\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^0\right) = \bar{\boldsymbol{J}}_i^0(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^0) = \bar{\boldsymbol{J}}_i^0 \boldsymbol{\tau}_i^0. \tag{3.9}$$

Since a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is used to model the factors, the energy of the factors, which is the norm of weighted residuals as defined in eq. (3.5), can be converted to the incremental information form $\mathcal{N}^{-1}(\boldsymbol{\eta}_i^0, \boldsymbol{\Lambda}_i^0)$.

$$E_i(t_i, \boldsymbol{\theta}_i) = \left\|\bar{\boldsymbol{r}}_i\right\|^2 = \boldsymbol{r}_i^\top(t_i, \boldsymbol{\theta}_i)\boldsymbol{\Lambda}_i \boldsymbol{r}_i(t_i, \boldsymbol{\theta}_i). \tag{3.10}$$

$$E_i(t_i, \boldsymbol{\tau}_i^0) \approx \frac{1}{2}\boldsymbol{\tau}_i^{0,\top}\boldsymbol{\Lambda}_i^0\boldsymbol{\tau}_i^0 - \boldsymbol{\tau}_i^{0,\top}\boldsymbol{\eta}_i^0 \quad \text{where} \tag{3.11}$$

$$\boldsymbol{\eta}_i^0 = -\bar{\boldsymbol{J}}_i^{0,\top}\bar{\boldsymbol{r}}_i^0 \quad \text{and} \quad \boldsymbol{\Lambda}_i^0 = \bar{\boldsymbol{J}}_i^{0,\top}\bar{\boldsymbol{J}}_i^0. \tag{3.12}$$

These representations of factors and the usage of the canonical form of Gaussian $\mathcal{N}^{-1}$ are vital for efficient conditioning and marginalization in the steps of the GBP algorithm, which is presented in the next section.

3.3 Gaussian Belief Propagation

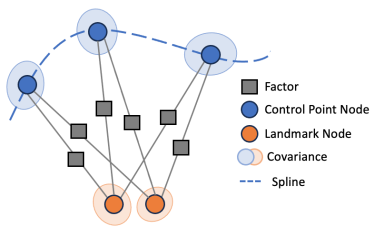


Figure 3.1: Factor Graph

As shown in fig. 3.1, SLAM scenarios can be represented as a factor graph, where sensor measurements become factors, and visual features and robot poses become nodes. Unlike other solving methods, Gaussian belief propagation considers the uncertainty of the nodes by assuming a Gaussian distribution. Thus, the covariance of each node is included in the factor graph, consisting of factors $f_i \sim \mathcal{N}^{-1}(\eta_{f_i}, \Lambda_{f_i})$ and nodes $n_j \sim \mathcal{N}(\mu_{n_j}, \Sigma_{n_j}) = \mathcal{N}^{-1}(\eta_{n_j}, \Lambda_{n_j})$. The canonical form of the Gaussian distribution is used to compute the product of probabilities easily. Gaussian belief propagation mainly consists of four steps to update each node and factor, and propagate the belief through iterative message passing. Details of each step are elaborated in the following.

3.3.1 Node Updates

First, the Gaussian nodes in the graph $n_j \in \mathcal{G}$ are updated through neighboring factors $f_i \in N(n_j)$ by taking a product over incoming factor-to-node messages $\boldsymbol{m}_{f_i \to n_j}$. Since we use the canonical form of Gaussians, where the information vector η lies in a vector space, we can convert the product to a simple summation by adding it to the prior node belief $P(n_j) = \mathcal{N}^{-1}(\boldsymbol{\eta}^p_{n_j}, \boldsymbol{\Lambda}^p_{n_j})$.

$$\boldsymbol{\eta}_{n_j} = \boldsymbol{\eta}^p_{n_j} + \sum_{f_i \in N(n_j)} \boldsymbol{\eta}_{f_i \to n_j} \quad \text{and} \quad \boldsymbol{\Lambda}_{n_j} = \boldsymbol{\Lambda}^p_{n_j} + \sum_{f_i \in N(n_j)} \boldsymbol{\Lambda}_{f_i \to n_j}. \tag{3.13}$$

The above equation can be perfectly applied in vector space states, such as landmark positions. However, for Lie groups G, which are extensively used in robotics for poses, the operators $\boxminus : G \times G \mapsto \mathfrak{g}$ and $\boxplus : G \times \mathfrak{g} \mapsto G$ rely on the tangent space $\mathfrak{g}$. Therefore, the above expression cannot be directly used for Lie groups. Instead, frame conversion and computing the increment of the factor-to-node messages on the tangent space are required to propagate all beliefs.
Thus, Lie group nodes follow the process proposed by Murai Murai et al. [15], using Mixed Gaussian Representation (MGR) by parameterizing it as $n_j \sim \mathfrak{N}(\boldsymbol{\mu}_{n_j}, \boldsymbol{\Lambda}_{n_j})$, where $\boldsymbol{\mu}_{n_j} \in G$ and $\boldsymbol{\Lambda}_{n_j} \in \mathbb{R}[\dim(\mathfrak{g}) \times \dim(\mathfrak{g})]$. The incoming message from a factor is also represented as $\boldsymbol{m}_{f_i \to n_j} \sim \mathfrak{N}(\boldsymbol{\mu}_{f_i \to n_j}, \boldsymbol{\Lambda}_{f_i \to n_j})$ to maintain the same form. In MGR formulation, the precision matrices are expressed in their associated elements in the Lie group. Consequently, the messages should be converted to the privileged frame $\boldsymbol{\mu}^0_{n_j}$ and $\boldsymbol{\Lambda}^0_{n_j}$, which represent the latest estimate of the node's state.

$$\boldsymbol{\tau}^0_{f_i \to n_j} = \boldsymbol{\mu}_{f_i \to n_j} \boxminus \boldsymbol{\mu}^0_{n_j} \quad \in \mathbb{R}^{\dim(\mathfrak{g})} \tag{3.14}$$

$$\boldsymbol{\Lambda}^0_{f_i \to n_j} = \left[\frac{\partial \boldsymbol{\tau}^0_{f_i \to n_j}}{\partial \boldsymbol{\mu}_{f_i \to n_j}}\right]^\top \boldsymbol{\Lambda}_{f_i \to n_j} \left[\frac{\partial \boldsymbol{\tau}^0_{f_i \to n_j}}{\partial \boldsymbol{\mu}_{f_i \to n_j}}\right] \quad \in \mathbb{R}^{\dim(\mathfrak{g}) \times \dim(\mathfrak{g})} \tag{3.15}$$

After converting the frame, the elements of $\boldsymbol{\tau}^0_{f_i \to n_j}$ and $\boldsymbol{\Lambda}^0_{f_i \to n_j}$ are in the tangent space relative to the privileged frame, which is a vector space. Therefore, the increments from messages can now be computed by summing in the same way as in eq. (3.13).

$$\boldsymbol{\tau}^+_{n_j} = \alpha_{n_j} \sum_{f_i \in N(n_j)} \boldsymbol{\Lambda}^0_{f_i \to n_j} \boldsymbol{\tau}^0_{f_i \to n_j} \quad \text{and} \quad \boldsymbol{\Lambda}^+_{n_j} = \sum_{f_i \in N(n_j)} \boldsymbol{\Lambda}^0_{f_i \to n_j}, \tag{3.16}$$

Where α_{n_j} denotes the step size. These increments are warped back to the updated frame, resulting in the updated state of the node.

$$\boldsymbol{\mu}_{n_j} = \boldsymbol{\mu}^0_{n_j} \boxplus \boldsymbol{\tau}^+_{n_j} \quad \text{and} \quad \boldsymbol{\Lambda}_{n_j} = \left[\frac{\partial \boldsymbol{\mu}_{n_j}}{\partial \boldsymbol{\tau}^+_{n_j}}\right]^\top \boldsymbol{\Lambda}^+_{n_j} \left[\frac{\partial \boldsymbol{\mu}_{n_j}}{\partial \boldsymbol{\tau}^+_{n_j}}\right]. \tag{3.17}$$

3.3.2 Node-to-factor Messages

The message from node to factor is generated analogously to eq. (3.16), differing only in that the increment from the target factor is excluded from the sums. However, since this process follows the node update, frame conversion to the latest state estimate must be carried out by reevaluating eq. (3.14) and eq. (3.15) first.

$$\boldsymbol{\tau}^{+}_{n_j \to f_k} = \sum_{f_i \in N(n_j)\setminus f_k} \boldsymbol{\tau}^{0}_{f_i \to n_j} \quad \text{and} \quad \boldsymbol{\Lambda}^{+}_{n_j \to f_k} = \sum_{f_i \in N(n_j)\setminus f_k} \boldsymbol{\Lambda}^{0}_{f_i \to n_j} \tag{3.18}$$

The outgoing message involves these values and their respective linearization point $\boldsymbol{\mu}^0_{n_j}$, by the triplet $(\boldsymbol{\mu}^0_{n_j}, \boldsymbol{\tau}^{+}_{n_j \to f_k}, \boldsymbol{\Lambda}^{+}_{n_j \to f_k})$.

3.3.3 Factor Updates

In analogy to the node update, the factor is updated by incoming messages from neighboring nodes $N(f_i)$. Before the update, the belief of the factor $B(f_i) = \mathcal{N}^{-1}(\boldsymbol{\eta}^0_{f_i}, \boldsymbol{\Lambda}^0_{f_i})$ is evaluated by eq. (3.12) from the cost function. So the intermediate quantities $\boldsymbol{\eta}'_{f_i}$ and $\boldsymbol{\Lambda}'_{f_i}$ are defined as

$$\boldsymbol{\eta}'_{f_i} = \boldsymbol{\eta}^0_{f_i} + \boldsymbol{\tau}^{+}_{N(f_i) \to f_i} \quad \text{and} \quad \boldsymbol{\Lambda}'_{f_i} = \boldsymbol{\Lambda}^0_{f_i} + \boldsymbol{\Lambda}^{+}_{N(f_i) \to f_i} \tag{3.19}$$

$\boldsymbol{\eta}^{+}_{N(f_i) \to f_i}$ and $\boldsymbol{\Lambda}^{+}_{N(f_i) \to f_i}$ denote the stacked vector and block diagonal matrix of the neighboring nodes' messages to the factor.

3.3.4 Factor-to-Node Messages

In this step, the factor is assumed to have a connection with two nodes. This approach can be similarly applied to a single connection or extended to more than two nodes. The updated states of the factor are as follows:

$$\boldsymbol{\eta}'_{f_i} = \begin{bmatrix} \boldsymbol{\eta}'_a \\ \boldsymbol{\eta}'_b \end{bmatrix} = \boldsymbol{\eta}^0_{f_i} + \begin{bmatrix} \boldsymbol{\eta}^{+}_{n_a \to f_i} \\ \boldsymbol{\eta}^{+}_{n_b \to f_i} \end{bmatrix} \tag{3.20}$$

$$\boldsymbol{\Lambda}'_{f_i} = \begin{bmatrix} \boldsymbol{\Lambda}'_{aa} & \boldsymbol{\Lambda}'^{\top}_{ba} \\ \boldsymbol{\Lambda}'_{ba} & \boldsymbol{\Lambda}'_{bb} \end{bmatrix} = \boldsymbol{\Lambda}^0_{f_i} + \begin{bmatrix} \boldsymbol{\Lambda}^{+}_{n_a \to f_i} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}^{+}_{n_b \to f_i} \end{bmatrix}. \tag{3.21}$$

To obtain a 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 should be marginalized by the Schur complement.

$$\boldsymbol{\eta}'_{f_i \to n_a} = \boldsymbol{\eta}^0_a - \boldsymbol{\Lambda}'^{\top}_{ba} \boldsymbol{\Lambda}'^{-1}_{bb} \boldsymbol{\eta}'_b \quad \text{and} \quad \boldsymbol{\Lambda}'_{f_i \to n_a} = \boldsymbol{\Lambda}^0_{aa} - \boldsymbol{\Lambda}'^{\top}_{ba} \boldsymbol{\Lambda}'^{-1}_{bb} \boldsymbol{\Lambda}'_{ba}. \tag{3.22}$$

In the same way, $\boldsymbol{m}_{f_i \to n_b}$ can also be obtained by permuting $\boldsymbol{\eta}'_{f_i}$ and $\boldsymbol{\Lambda}'_{f_i}$ to marginalize n_a.

$$\begin{bmatrix} \boldsymbol{\eta}'_b \\ \boldsymbol{\eta}'_a \end{bmatrix} = P \boldsymbol{\eta}'_{f_i} \text{ and } \begin{bmatrix} \boldsymbol{\Lambda}'_{bb} & \boldsymbol{\Lambda}'^{\top}_{ab} \\ \boldsymbol{\Lambda}'_{ab} & \boldsymbol{\Lambda}'_{aa} \end{bmatrix} = P \boldsymbol{\Lambda}'_{f_i} P^{\top}, \tag{3.23}$$

In analogy to eq. (3.17), the incremental values $\boldsymbol{\eta}'_{f_i \to n_a}$ and $\boldsymbol{\Lambda}'_{f_i \to n_a}$ should be converted into the updated outgoing frame of reference, which becomes a factor-to-node message $\boldsymbol{m}_{f_i \to n_a}$. Step size α_{f_i} is used in this case as well.

$$\boldsymbol{\tau}'_{f_i \to n_a} = \alpha_{f_i} \boldsymbol{\Lambda}'^{-1}_{f_i \to n_a} \boldsymbol{\eta}'_{f_i \to n_a} \in \mathbb{R}^{\dim(\mathfrak{g})}, \quad \boldsymbol{\mu}_{f_i \to n_a} = \boldsymbol{\mu}^0_{n_a} \boxplus \boldsymbol{\tau}'_{f_i \to n_a} \in G \tag{3.24}$$

$$\text{and} \quad \boldsymbol{\Lambda}_{f_i \to n_a} = \left[\frac{\partial \boldsymbol{\mu}_{f_i \to n_a}}{\partial \boldsymbol{\tau}'_{f_i \to n_a}} \right]^{\top} \boldsymbol{\Lambda}'_{f_i \to n_a} \left[\frac{\partial \boldsymbol{\mu}_{f_i \to n_a}}{\partial \boldsymbol{\tau}'_{f_i \to n_a}} \right] \in \mathbb{R}^{\dim(\mathfrak{g}) \times \dim(\mathfrak{g})}. \tag{3.25}$$

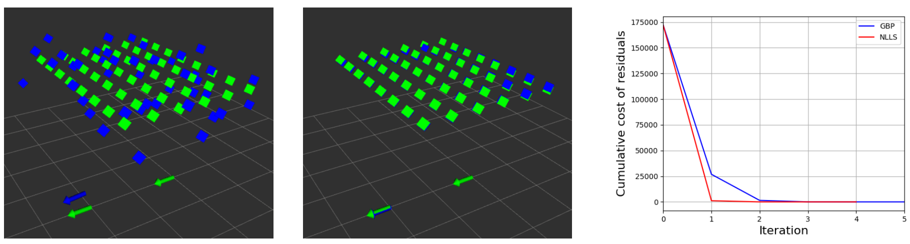
Chapter 4

Experiment

In this project, the framework for the GBP algorithm with continuous time formulation, Hyperion [16], was used. Initially, there was a misapplication in the node update step, where the increment τ and information vector η for vector space nodes were incorrectly converted. Only the $\mathbb{SE}(3)$ nodes functioned correctly. This issue was resolved during the project and verified using both simulation and real-world datasets. To compare the GBP solver against the centralized NLLS solver, the established framework, Ceres, was used.

4.1 Simulation

The GBP algorithm for continuous time representation of visual features was verified by simulating a triangulation scenario using stereo poses. In this simulation, visual features were represented by a 6x4 grid of board landmarks. Initially, both the landmarks and one of the stereo poses were perturbed. GBP was then applied to minimize the residual between the pixel measurements and the projected pixels from the estimated landmarks. To compare the performance of GBP with the NLLS solver, synchronized data updates were used, and the tests were conducted with NLLS as well. Both the landmarks and the pose were initialized with identical perturbations of 0.5 meters in translation and 0.5 radians in rotation. This setup allowed for a thorough comparison of the two methods under controlled conditions.



(a) Initial Perturbed Setup (b) Converged Result (c) Convergence

Figure 4.1: Estimated pose(arrow) and landmarks(square) in simulation with a 0.5 m/rad perturbation. GBP(blue) converges into the ground truth(green).

As the residuals of all factors, defined by eq. (3.11), are accumulated in each iteration, the convergence can be assessed by evaluating the cumulative cost of these residuals. The simulation results indicate that both the GBP and NLLS solvers converge to the same optimal solution, even under high perturbation. However, since GBP estimates uncertainty using the covariance of the nodes—an aspect not considered by NLLS—the convergence rate of GBP is slower compared to NLLS.

4.2 Dataset

To validate the algorithm with a real-world dataset, three videos featuring a 6x6 AprilTag grid board were recorded using a ZED X stereo camera. The ground truth of the $\mathbb{SE}(3)$ camera pose was obtained from the detected AprilTag corners, which lie on the same plane, using the Perspective-n-Point (PnP) algorithm. Additionally, the ground truth positions of the AprilTag corners, considered as landmarks, were extracted using Bundle Adjustment (BA) across all video frames. The evaluation metric employed is the Root Mean Square Error (RMSE) in both translation and orientation of the estimated poses. Since the data was captured with a stereo camera, to assess the convergence of the non-linear optimization solver, the poses and landmarks were perturbed, while the first frame poses were kept fixed.

4.3 Discrete Time

Before testing the GBP algorithm with the continuous time formulation, the algorithm was first evaluated in discrete time to verify the dataset. Since GBP estimates the uncertainty of the nodes, the initial covariance for each node must be initialized appropriately. To determine the optimal value for the initial covariance, various setups were tested across three datasets. The step sizes during node updates α_{n_j} and factor updates α_{f_i} were set to 0.8, and a perturbation of 0.1 m/rad was applied to both the poses and landmarks.

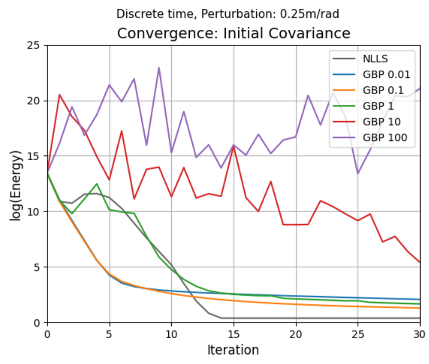


Figure 4.2: Simulation in Discrete time GBP setup with initial covariance ranging from 0.01 to 100 times the identity matrix. Convergence through iterations is shown by cumulative cost of residuals, as defined in eq. (3.11).

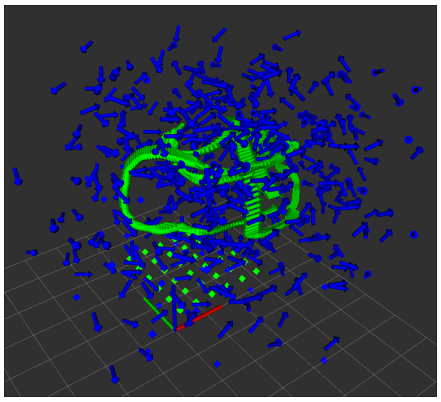
The convergence status and rate, as shown in fig. 4.4, can vary depending on the initial covariance. Both the NLLS solver and GBP with identity initial covariance exhibited a similar trend: they initially increased slightly before eventually converging. In contrast, using an initial covariance scaled by 0.1 led to convergence without any fluctuations. Since the step sizes for node updates and factor updates in GBP are fixed, GBP converges more slowly to the optimal solution compared to NLLS, which converges rapidly. The estimation of covariance in GBP also contributes to its slower convergence rate.
GBP has now been tested under various perturbation setups with an initial covariance of 0.1 times the identity matrix. Three datasets were used in this experiment,

and the accumulated RMSEs of the optimized poses were compared to those obtained using the NLLS solver, as shown below.

Table 4.1: Root Mean Square Errors (RMSEs) in rotation and translation resulting from GBP and NLLS solvers under different perturbation levels across three datasets. The solver runs to converge or terminate after 50 iterations.

	Perturbation	**Scene 1**		**Scene 2**		**Scene 3**	
	[m/rad]	**GBP**	**NLLS**	**GBP**	**NLLS**	**GBP**	**NLLS**
Rotation [rad]	0.1	0.040	0.039	0.048	0.048	0.046	0.046
	0.25	0.042	0.039	0.049	0.049	0.046	0.046
	0.5	0.042	0.568	0.049	0.497	0.046	0.418
Translation [m]	0.1	0.033	0.034	0.034	0.034	0.042	0.042
	0.25	0.041	0.034	0.034	0.034	0.042	0.042
	0.5	0.047	1.466	0.034	0.590	0.042	0.537

For all datasets, GBP successfully converged with perturbations up to 0.5 m/rad, whereas the NLLS solver diverged in all cases. The initial perturbed setup and the results after 50 iterations are visualized in fig. 4.3. While the optimized poses and landmarks diverged in NLLS, GBP consistently produced results that were closer to the ground truth.


(a) Initial Perturbed Setup

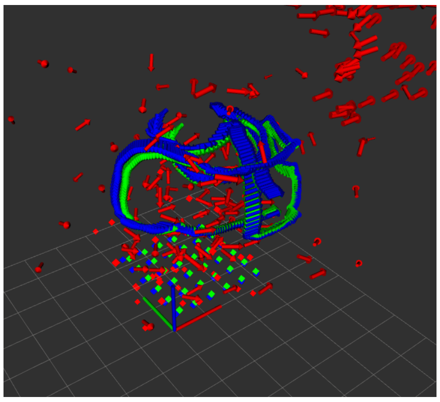

(b) Optimized Result

Figure 4.3: Estimated pose(arrow) and landmarks(square) for Scene 1 with a 0.5 m/rad perturbation in discrete time. The optimized result after 50 iterations shows GBP(blue) converges to the ground truth(green) while NLLS(red) diverges.

4.4 Continuous Time

For the continuous time formulation, a 4-degree-of-freedom Z-spline was used. The control points of the spline were spaced at 200 ms intervals, corresponding to a data sampling rate of 10 Hz. As in the discrete time approach, the initial covariance in GBP was tested first. The step sizes for node updates and factor updates were set to 0.4 and 0.7, respectively.
Similar to the discrete time case, using an initial covariance of either the identity matrix or 0.1 times the identity matrix resulted in stable convergence. However, NLLS and some instances of GBP exhibited fluctuations near the optimal solution. Since each pose is represented by four control point nodes, all factors are connected to these nodes, complicating the graph structure with additional optimizable pa-

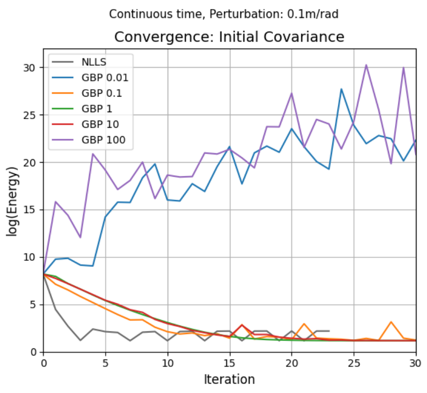
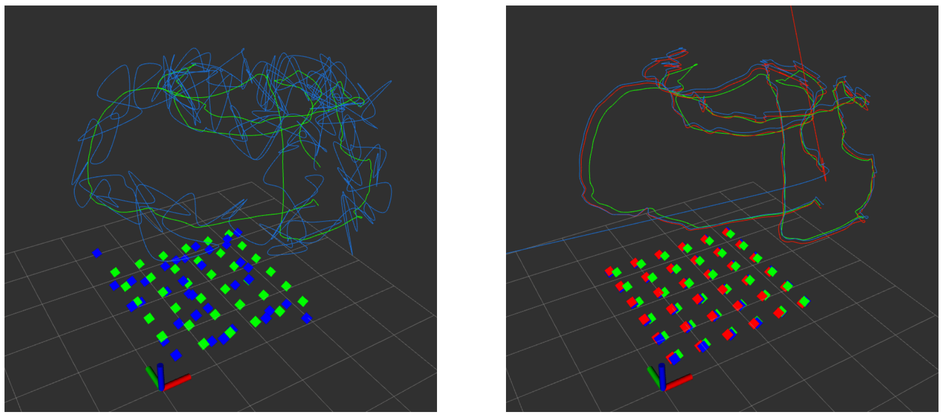


Figure 4.4: Convergence of the continuous time GBP setup with initial covariance ranging from 0.01 to 100 times the identity matrix.

rameters. The initial covariance with the identity matrix was used for testing with perturbations.
All three datasets were tested with various perturbation setups for GBP and NLLS. Only the 0.1 m/rad perturbation in Scene 1 achieved convergence, while the other perturbation levels led to divergence in both methods. In Scene 1, landmarks were uniformly captured by both stereo cameras across all frames. In contrast, Scenes 2 and 3 involved rapid movements, leading to dropout of landmark measurements.



(a) Initial Perturbed Setup (b) Converged Result

Figure 4.5: Estimated pose(line) and landmarks(square) of Scene 1 with a 0.1 m/rad perturbation in continuous time. The optimized results after 50 iterations show that GBP(blue) and NLLS(red) converge to the ground truth(green).

As shown in fig. 4.5, the converged trajectory defined by the spline intuitively represents the camera's motion in continuous time. The landmarks also converged to the grid-aligned ground truth.

Table 4.2: Root Mean Square Errors (RMSEs) in rotation and translation resulting from GBP and NLLS solver with a 0.1 m/rad perturbation in Scene 1.

	Continuous Time		**Discrete Time**	
	GBP	**NLLS**	**GBP**	**NLLS**
Rotation [rad]	0.044	0.041	0.040	0.0390
Translation [m]	0.045	0.038	0.033	0.034

To compare the results of the continuous time and discrete time setups under the same perturbation of 0.1 m/rad, the RMSEs for the discrete time formulation were lower than those for the continuous time formulation in both rotation and translation errors. Since the dataset consisted of synchronized stereo images, the advantage of the continuous time representation designed to handle high-rate or asynchronous data did not become evident in this case.

Chapter 5

Conclusion

In this project, the GBP framework with continuous time representation using Z-splines has been validated using a dataset featuring an AprilTag grid board. The effectiveness of our method, which leverages visual features, was demonstrated in comparison to a conventional NLLS solver, showing superior robustness to perturbations, particularly in discrete time formulations. Since GBP estimates the uncertainty of the nodes, the initial covariance can significantly influence both the convergence rate and the stability of the optimization process. As the number of nodes, defined by control points that shape the spline motion, increases, the complexity of the graph grows, making the optimization problem more challenging to solve.

To enhance the stability of the GBP framework in continuous time, further analysis is required on hyperparameters such as the sampling time of control points and step sizes during node and factor updates. Implementing and testing the method in a distributed system with asynchronous data will also be crucial to fully demonstrate its effectiveness and robustness in real-world applications.

Bibliography

[1] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *arXiv preprint arXiv:1901.03642*, 2019.

[2] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[3] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2018, pp. 5000–5007.

[4] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Robotics: Science and Systems XI*, 2015.

[5] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *2012 IEEE International Conference on Robotics and Automation.* IEEE, 2012, pp. 2088–2095.

[6] J. Ortiz, T. Evans, and A. J. Davison, "A visual introduction to gaussian belief propagation," *arXiv preprint arXiv:2107.02308*, 2021.

[7] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE transactions on robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[8] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[9] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[10] S. Anderson, F. Dellaert, and T. D. Barfoot, "A hierarchical wavelet decomposition for continuous-time slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2014, pp. 373–380.

[11] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras." in *BMVC*, vol. 2, no. 5, 2013, p. 8.

[12] Z. Peng, Y. Xu, M. Yan, and W. Yin, "Arock: an algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2851–A2879, 2016.

[13] P. Bänninger, I. Alzugaray, M. Karrer, and M. Chli, "Cross-agent relocalization for decentralized collaborative slam," in *2023 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2023, pp. 5551–5557.

[14] A. J. Davison and J. Ortiz, "Futuremapping 2: Gaussian belief propagation for spatial ai," *arXiv preprint arXiv:1910.14139*, 2019.

[15] R. Murai, J. Ortiz, S. Saeedi, P. H. Kelly, and A. J. Davison, "A robot web for distributed many-device localisation," *IEEE Transactions on Robotics*, 2023.

[16] D. Hug, I. Alzugaray, and M. Chli, "Hyperion-a fast, versatile symbolic gaussian belief propagation framework for continuous-time slam," *arXiv preprint arXiv:2407.07074*, 2024.

[17] J. T. Becerra-Sagredo, "Z-splines: moment conserving cardinal spline interpolation of compact support for arbitrarily spaced data," *SAM Research Report*, vol. 2003, 2003.