POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Multi-robot localization: Gaussian belief propagation on factor graph

Supervisors

Prof. Marcello CHIABERGE

Dott. Mauro MARTINI

Candidate

Giorgio AUDRITO

DECEMBER 2024

Summary

The development of multi-robot swarm technology represents a transformative frontier in robotics, offering vast potential across industries such as disaster response, environmental monitoring, logistics, and large-scale agricultural operations. In this scenario, the ability of robots to effectively localize themselves becomes paramount. However, traditional centralized methods quickly become impractical, suffering from communication bottlenecks, limited scalability, and vulnerability to system failures. This challenge underscores the need for distributed solutions, where each robot independently contributes to the overall system, ensuring robust and scalable performance. The aim of this thesis is to investigate the capability of factor graphs to model the complexities of multi-robot systems and examine how Gaussian Belief Propagation (GBP) enables efficient inference on such models. Specifically, this research focuses on applying these methods to the problem of multi-robot localization, achieved through the fusion of odometry and ultra-wideband (UWB) measurements in a decentralized robotic swarm. Hence, this work aims to provide a robust, scalable solution for accurate localization in large-scale autonomous systems. The performance of the proposed solution has been evaluated through both simulations and real-world experiments. An ablation study is performed in simulation to study the scalability of the solution increasing the number of robots and the noise. Additionally, real-world testing will be conducted using a swarm of four TurtleBot3 robots, with the Vicon motion capture system serving as the ground truth reference. The system effectively integrates asynchronous data from ultra-wideband (UWB) signals, odometry, and inter-robot positional information, achieving a global localization error of under 15 cm in real-world experiments. The solution has proven scalable from single-robot setups to fleets of up to one hundred.

Table of Contents

List of Tables

List of Figures

Acronyms

GBP
Gaussian Belief Propagation

UWB
Ultra Wide Band

iSAM
Incremental Smoothing and Mapping

SLAM
Simultaneous Localization And Mapping

PDF
Probability density function

BP
Belief Propagation

GPR
Gaussian Process Regression

RMSE
Root mean square deviation

MAE
Mean absolute error

ROS2
Robot Operating System 2

Chapter 1

Introduction

1.1 Objectives of the thesis

For over a century, society has been captivated by the promise of autonomous systems, envisioning a future where machines perform complex tasks, adapting, learning, and responding to their environments. Nowadays, this vision is evident in robotics, where improvements have pushed robots beyond industrial assembly lines and into domains once devoted to human expertise. From healthcare to disaster response, robots are being integrated into an ever-growing array of fields, sparking excitement about their potential to transform the world as we know it.
However, as robots become more deeply embedded in our lives, the need for cooperative systems, where robots collaborate as unified swarms or communities, becomes increasingly urgent. These systems hold the power to accomplish tasks that individual robots alone could not achieve, raising profound questions and presenting both significant challenges and opportunities.
The goal of managing vast swarms of robots conceals the need to develop decentralized systems that can overcome the problems of traditional centralized techniques, which very quickly become impracticable, as they are unable to scale effectively on a large number of robots, present problems related to the exchange of large amounts of data with the computing unit that coordinates them, and are inherently more susceptible to attacks and system failures. Applying such swarms of robots in increasingly complex scenarios requires the ability to model complex environments and systems and merge heterogeneous and asynchronous information, all while having the limited computation capability of mobile robots.
This thesis aims to investigate factor graphs as a technique for modeling complex systems such as swarms of robots and the use of the Gaussian Belief Propagation (GBP) algorithm to make inferences about the factor-graph modeling of the system. Specifically, the thesis addresses the problem of localizing a decentralized swarm of

robots using Ultra Wide Band antenna (UWB) and odometry. The development of a factor graph model that effectively described the problem and allowed for an efficient implementation of the GBP to fully utilize the CPU power of the account to make the algorithm executable in real time was a key achievement. The algorithm's performance was evaluated in simulation to determine the algorithm's scalability and ability to effectively fuse information from the two sensors. The algorithm was also tested in a real scenario using a swarm of four TurtleBot3 and six UWB anchors and a Vicon vision system as reference, demonstrating the algorithm's effectiveness within a real scenario.

This project originated at the PIC4SeR (PoliTo Interdepartmental Center for Service Robotics) as part of a broader initiative dedicated to advancing service robotics. The aim is to develop cutting-edge solutions across diverse fields, including precision agriculture, smart cities, healthcare, cultural heritage preservation, and space exploration. Within this context, the ability to achieve fully decentralized localization for a swarm of robots, enabling asynchronous data sharing, is a critical requirement for realizing real swarm autonomy.

1.1.1 Thesis Structure

The structure of the thesis is as follows:

1. Chapter 2 provides an overview of the state of the art in factor graphs and the optimization algorithms used for inference.
2. Chapter 3 presents the theoretical foundations of factor graphs and the GBP algorithm.
3. Chapter 4 explains the implementation of the proposed solution.
4. Chapter 5 describes the experimental setup used to evaluate the solution.
5. Chapter 6 presents the performance results of the proposed solution.
6. Chapter 7 concludes the work, highlighting key findings and suggesting potential avenues for future research.

Chapter 2

State of the Art

The ability to accurately determine robot's location is essential to enabling autonomous robotic systems. Over the years, numerous techniques have been developed to address this challenge. This chapter reviews key contributions to localization using factor graphs and GBP.

2.1 Localization with factor graph

A key milestone in robotics research is the application of factor graphs, as detailed in the constitutional work of Dellaert and Kaess [1]. This study highlights the ability of factor graphs to simplify and analyze complex inference problems. These graphs provide a robust framework for designing advanced solutions and performing efficient inference on constructed models. The authors demonstrate how nonlinear optimization techniques can be used to solve general nonlinear factor graphs, taking advantage of the system's sparsity to improve computational efficiency. They also extend their approach to optimization on nonlinear manifolds, further enhancing its applicability.

The work emphasizes the importance of sparsity in the solution process, exploring how the structure of a solution influences this property and how sparsity can be exploited to create high-performance algorithms. A key innovation presented in this study is iSAM [2] (Incremental Smoothing and Mapping), an algorithm that employs the Bayes Tree data structure to reuse prior computations efficiently and optimize graphical models. The iSAM [2] framework was later improved in subsequent works, such as iSAM2 [3] and MR-iSAM2 [4]. These contributions show how SLAM (Simultaneous Localization And Mapping) problems can be effectively modeled using factor graphs, as shown in Figure , demonstrating the efficiency

of the proposed optimization techniques in solving real-world robotics challenges. Together, these works have become foundational in the field, inspiring numerous studies and establishing a significant portion of the current state of the art in SLAM.

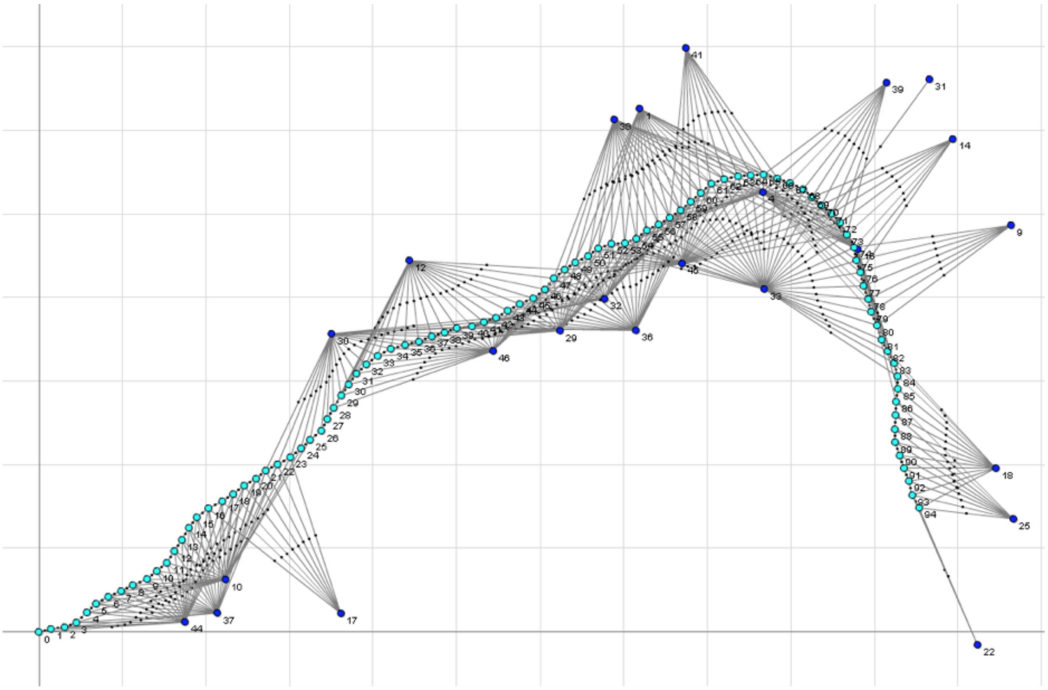


Figure 2.1: Factor graph for SLAM (Frank Dellaert and Michael Kaess)[1]

In multi-robot systems, factor graphs have been widely used to model localization problems, with many solutions relying on centralized pose graph optimization. One notable example in this domain is the work conduct by Andersson et al. C-SAM [5]. This study introduces a framework for collaborative simultaneous localization and ,mapping, employing square root information smoothing to allow multiple robots to construct and merge maps without prior knowledge of their relative positions. In this framework, factor graphs play a critical role in representing and optimizing the trajectories of multiple robots. By integrating odometry and rendezvous measurements, the pose graph creates a robust foundation for merging maps and recovering accurate trajectories. This approach demonstrates how factor graphs can facilitate collaborative mapping in complex multi-robot scenarios.

The multi-robot problem was also addressed by the work of Indelman et al.[6], who handled the challenges of multi-robot localization and data association when the robots' initial relative positions were unknown. The authors propose an expectation-maximization algorithm to infer initial relative poses and resolve data

association issues. Factor graphs are used to model the poses of multiple robots and the sensor measurements linking them. In this representation, nodes correspond to robot poses at specific times, while factors encode relative transformations derived from sensor data. Synthetic and real-world experiments validating this method highlight its robustness to outliers and ability to accurately infer initial relative poses, even under erroneous starting conditions. Further advancing the field is provided by Been Kim et al.[7], who introduce a novel approach for cooperative SLAM involving multiple robots. This work extends the iSAM [2] framework to handle multiple pose graphs, improving the efficiency and robustness of multi-robot mapping. By formulating relationships between pose graphs in a fashion that reduces computational complexity, the proposed method avoids the initialization challenges typically associated with global formulations. The authors evaluate the performance of their solution using a publicly available multi-robot laser dataset and data collected collaboratively by a helicopter and a ground robot. The results demonstrate faster convergence rates and improved computational efficiency compared to global parameterization methods.

Despite the effectiveness of these centralized approaches, they rely on a base station to aggregate and process data, which can create vulnerabilities, demand high communication bandwidth, and limit scalability. To address these limitations, decentralized solutions have gained attention. Approaches such as those described in Trevor Halsted et al.[8] survey demonstrate the feasibility of achieving centralized-equivalent localization in decentralized settings. These methods highlight the potential for robust and scalable multi-robot localization, even when communication networks are sparse or never fully connected.

2.2 Optimization on factor graph

Traditionally, the graph optimization problem is formulated as a linear minimum mean-squared-error problem and solved using iterative optimization techniques, such as the Gauss-Seidel or Jacobi methods. For example, the Jacobi method is applied in the work of R. Aragues et al. [9]. To manage a heterogeneous swarm of robots effectively, it is essential to implement appropriate strategies and techniques; the system must be capable of handling heterogeneous, asynchronous, and noisy data closely tied to the optimization algorithms applied to the factor graph. Several algorithms have been proposed to address these challenges, such as iSAM [2] by Dellaert and the approach outlined in the recent work by Tian et al.[10]. The latter utilizes block coordinate descent on Riemannian manifolds and is later employed in Kimera-Multi[11].

While some of these methods adopt a distributed approach, as seen in the work of Siddharth Choudhary et al.[12], which employs a distributed Gauss-Seidel method capable of scaling to systems with up to fifty robots, they exhibit a critical limitation. These synchronous methods require data from different sensors and robots to arrive at predetermined intervals. This constraint significantly affects their scalability and real-world applicability in dynamic, asynchronous environments.

The first asynchronous algorithm for distributed pose graph optimization was proposed in the work of Yulun Tian et al. [13], which proposed a solution that allowed asynchronous stochastic parallel pose graph optimization by computing gradient descendent on a Riemannian manifold in a distributed fashion.
An emerging and effective technique is the GBP algorithm, which converges with asynchronous message passing and allows general nonlinear sensor models and robust factors. It is capable of handling a large number of non-Gaussian outlier measurements. A reference work for deploying the GBP algorithm on a factor graph is the work of Andrew J. Davison and Joseph Ortiz [14]. This paper highlights GBP's efficiency in leveraging distributed processing and storage, its flexibility to adapt to various estimation problems, and its suitability for real-time applications due to its alignment with modern processor architectures. This paper effectively shows the deployment of GBP on several problems ranging from imaging denoising to SLAM; relevant contributions are provided with the integration of the M-Estimator inside the Gaussian factor. Also, the work of Joseph Ortiz et al. [15] is a handy paper for understanding the deployment of GBP on factor graphs. From this work, several relevant works emerged. A thoughtful work, Robot Web [16] of Riku Murai provides contributions in order to create a framework based on GBP for robot localization design for scalable and dynamic swarm with robots that join and leave the network at will based on a web page, and it provides an extension of the GBP formulation to support Lie Groups. The solution is tested with simulation and real-world experiments with nine physical robots.
An exciting application of the GBP algorithm is in the work of Aalok Patwardhan et al. [17], who provide a distributed framework based on GBP design to coordinate and plan robot trajectory over a forward time window. The proposed framework has shown effectiveness in simulation, even in communication failure scenarios, and planning and coordination performance, which overcame traditional state-of-the-art techniques. Another interesting paper is the work of Riku Murai [18], which is built on top of Robot Web [16] to provide a novel method for multi-robot localization through an extrinsic sensor and marker calibration that allows not only the localization of the robots in the swarm but also to calibrate independently each robot sensors specific characteristic providing a handy tool for real word robot deployments, it also extends the previous work of Robot Web form SE(2) to SE(3).

In conclusion, exploring factor graphs and GBP highlights their significant contributions to the multi-robot localization problem. The fundamental innovative approaches, such as extending the iSAM framework for multiple pose graphs, as discussed in the work of Been Kim et al.[7], and the application of GBP for distributed optimization, as seen in the work of Andrew J. Davison and Joseph Ortiz [14], demonstrate their efficiency, scalability, and robustness. Factor graphs effectively model complex relationships in robotic systems, while GBP provides a flexible and efficient algorithm for real-time probabilistic estimation. The proposed works highlight that factor graphs and GBP are pivotal in advancing robotic localization and invite ongoing exploration and innovation in this exciting field.

Chapter 3

Theory Background

In a world increasingly shaped by intelligent machines and autonomous systems, the ability to make decisions under uncertainty has become a defining challenge. From self-driving cars navigating busy streets to robots interpreting complex environments, these systems must process vast amounts of data in real-time, often without complete information.
At the core of this capability lies Bayesian probabilistic inference, a mathematical framework that allows machines to update their understanding as new information becomes available. Unlike traditional methods that rely on fixed assumptions, Bayesian inference offers a flexible and dynamic approach to decision-making, continuously refining predictions based on evolving data.

As robotics evolve toward more dynamic and heterogeneous system representations capable of handling complex abstractions, Bayesian inference presents computational challenges. While various algorithms can effectively infer using problem-specific structures, the inference on dynamic and complex systems often proves computationally intensive. This has emerged as a limiting factor for systems that require real-time processing.
In this evolving scenario, scalable Bayesian inference is imperative. To scale effectively and fully leverage available computation across different architectures, it is essential to develop inference methods capable of operating with distributed local processing, storage, and message-passing communication without requiring a global view or coordination of the entire model.Recent advances, such as MCMC (Markov Chain Monte Carlo), variational inference, and factor graphs, have helped address some scalability challenges.

The solution proposed in this thesis relies on Factor as a structured way to represent complex models by breaking them into manageable parts and Loopy Belief Propagation that has proven effectiveness in solving large probabilistic models,

especially when dealing with continuous variables under Gaussian assumptions [19].

3.1 Factor graph

Factor graphs are closely related to other PGMs (Probabilistic Graphical Models), particularly Bayesian networks and Markov random fields.
While Bayesian networks use directed acyclic graphs to represent conditional dependencies between variables, and Markov random fields use undirected graphs to represent dependencies, factor graphs provide a more flexible representation that can incorporate both directed and undirected edges. Furthermore, both Bayesian networks and Markov random fields can be converted into factor graphs, making factor graphs a unifying framework for representing complex systems.

The bipartite nature of factor graphs distinguishes them from other graphical models. In a bipartite graph, nodes are divided into two disjoint sets such that no two nodes within the same set are adjacent. In the context of factor graphs, these two sets are the variable nodes and the factor nodes. The variables are numerical parameters of a system whose values we wish to estimate, but are not directly observable, typically denoted as X_i. While factor node represents local function, which defines a specific relationship or constraint involving a subset of variables. Each factor function $f_i(S_i)$ relates to a set of variable nodes and encapsulates the dependency structure by quantifying how the variables interact to contribute to the overall system's behavior.

In a factor graph, a factor is defined by three key components: a function of the hidden variables, a measured value, and a PDF (probability density function).

1. The function associated with a factor describes the mathematical relationship or dependency between the subset of variables it connects, $f_i(S_i)$. This function is usually local and contributes to the factorization of the global objective, such as a joint probability distribution.

2. The measured value z_s represents the observed data or known values within the system, anchoring the function's behavior to ensure that the factor accurately reflects empirical data or predefined system constraints.

3. The PDF (probability density function) expresses the factor's probabilistic nature, indicating the likelihood of different variable configurations based on the measured values and the function's relationship. The PDF plays a crucial role in modeling uncertainty and variability within the system, allowing the factor to quantify how various combinations of variables affect the overall distribution.

These three components (function, measured value, and PDF) enable the factor to capture deterministic and probabilistic relationships within the model, offering a comprehensive representation of the system's structure.

3.1.1 Factor Graph in localization

Factor graphs in the context of robot localization provide a structured representation for relating observations, motion estimates, and map data. The nodes in these graphs represent unknown robot states or sensor observations, and the factors encapsulate probabilistic relationships between these states, often modeled as Gaussian distributions. This factorization enables a modular approach to localization, where each sensor measurement, motion model, or environmental landmark is treated as an individual factor, allowing factor graphs to seamlessly integrate multiple sensors such as lidar, GPS, and inertial measurements, enhancing robustness and resilience against individual sensor failures. The inherent flexibility of factor graphs makes them suitable for different localization tasks, including static and dynamic environments, in indoor and outdoor settings.

In order to localize the robot, a factor graph that is properly capable of modeling the localization problem is needed, and an example is provided in the Figure 3.1.

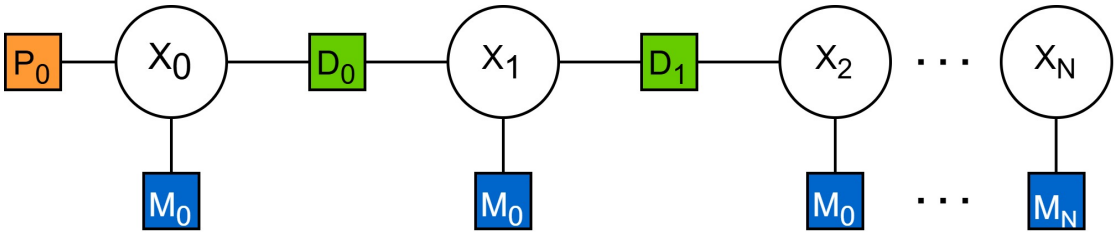


Figure 3.1: Basic factor graph for localization

The factor graph shown in the Figure 3.1 is an essential representation of how a localization problem can be modeled using a factor graph.
Each circle denotes an unknown variable, which, in this example, corresponds to the robot's pose at a specific time instant. The green squares, labeled as D_n, are factors that connect the current pose to the previous one. These factors encode internal knowledge about changes in the robot's pose, such as its dynamics or measurements from inertial or odometry sensors. The square labeled P_0 is a prior factor linked only to the variable representing the robot's initial position. It encodes our prior knowledge about the robot's starting position. The final type of factor, labeled M_n, corresponds to external measurements. In this example, M_n indicates that an external measurement is available at each instant to determine the robot's location.

In summary, the factor graph models the robot's movement, starting from a known initial position. It updates the robot's position at each time step by incorporating internal and external data.

The provided example is just a first approach to the problem; in order to properly model a localization problem, many factor graph structures could be explored. An example is shown in the Figure 3.2.

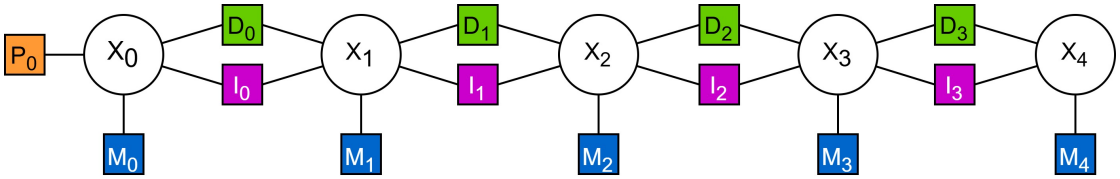


Figure 3.2: Factor graph with inertial and odometry sensor fusion

The factor graph shown in the Figure 3.2 is designed to combine two different types of internal robot measurements; for example, the robot's state change is determined based on data from both odometry and the inertial sensor. This fusion enhances the robot's ability to handle noise during localization.
It is important to note that if a set of factors is connected to the same set of variables, these factors can be merged into a single factor. This merging simplifies the graph model, reduces loopiness, and improves the sparsity of the connectivity matrix. This leads to better convergence for optimization algorithms, such as belief propagation. However, this choice may also introduce more complex and potentially non-linear factors, which can be challenging to manage.

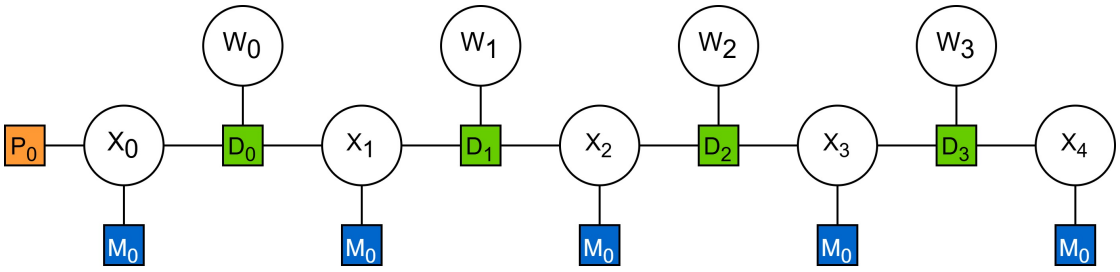


Figure 3.3: Factor graph with parameter auto-calibration

Another type of structure, illustrated in the Figure 3.3, is a simplified version of the model developed by Riku Murain[18]. In this model, the objective is to estimate the positions of robots within a swarm and determine specific parameters unique to each robot. These parameters cannot be predefined during the design phase and may be dynamic.
In the Figure 3.3, these parameters are represented by the W_n variables. Each of

these variables is linked to a factor, such as an odometry factor, which estimates a parameter to correlate the positions of two robots better. For example, this parameter could account for slippage during the robot's movement in an odometry motion model.

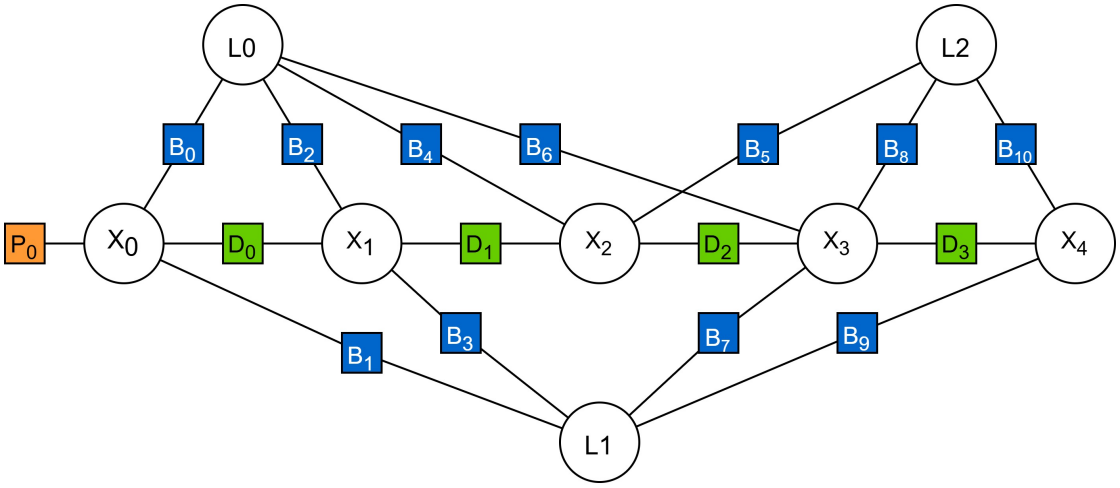


Figure 3.4: Factor graph for landmark based localization

A standard structure, illustrated in the Figure3.4, s the same structure proposed in the work of Dellaert[1]. This structure represents a localization problem where the robot determines its position by relying on landmark measurements taken at different relocation instances. One example of this approach is visual odometry, in which the robot's motion is estimated based on observed features from various viewpoints. In this context, the graph connects each X_n to a set of landmark variables using measurements, such as bearing measurements. This type of structure is widely used in robotics, as it relies on a sparse matrix representation of the system, which enables fast and efficient optimization.

3.2 Belief Propagation

Gaussian Belief Propagation is an advanced inference algorithm that estimates variables in systems represented by graphical models. It extends the classic BP (Belief Propagation) algorithm, specifically adapting it for scenarios where variables follow Gaussian distributions. This adaptation takes advantage of the characteristics of Gaussian distributions, allowing for more efficient and accurate inferences.
The classic BP algorithm is widely used in fields such as computer vision, error correction, and artificial intelligence, and it works well with discrete variables. However, it faces challenges when dealing with continuous variables, particularly

Gaussian ones. GBP addresses these challenges by incorporating Gaussian assumptions, making it applicable to a broader range of real-world problems.

This chapter explores GBP's theoretical foundations, including its mathematical basis and practical implementations. We will examine how GBP modifies BP's message-passing framework to handle Gaussian variables and discuss its advantages over traditional methods.

3.2.1 Overview of Belief Propagation

Belief Propagation is a message-passing technique employed for inference on graphical models. Its primary function is determining the marginal distributions of latent variables based on observed data.
Traditional BP is designed for discrete variables, which limits its applicability in continuous domains. To address this limitation, GBP extends the BP framework to accommodate continuous variables represented by Gaussian distributions.

To introduce the BP algorithm, two statements must be made. The first is that the factor graph is a tree structure, and the probability distribution could be of any type, discreet, or continuous. The provided derivation will follow the notation given in Bishop's book Pattern Recognition and Machine Learning [20].

$$p(x) = \prod_i f_i(X_i) \tag{3.1}$$

The equation 3.1 defines the probability distribution over all variables in a factor graph as a product of all factors; in order to perform inference of the variables of interest, there is the need to compute the marginal distribution of those variables, choosing one variable x, its marginal distribution is found by taking the joint distribution, and summing over all of the other variables:

$$p(x) = \sum_{X \setminus x} p(x) \tag{3.2}$$

As shown in equation 3.2, here the notation $X \setminus x$ means all elements of X except x.

Looking at the Figure 3.5, which highlights an arbitrary variable x within a tree factor graph, it is possible to see that x is directly connected to several factor f_s, while every other factor in the graph is connected to x indirectly via exactly one of these factors, so is possible to divide the whole graph into the same number of subsets as the factors f_s, and write the whole joint probability distribution as a product of these subsets:

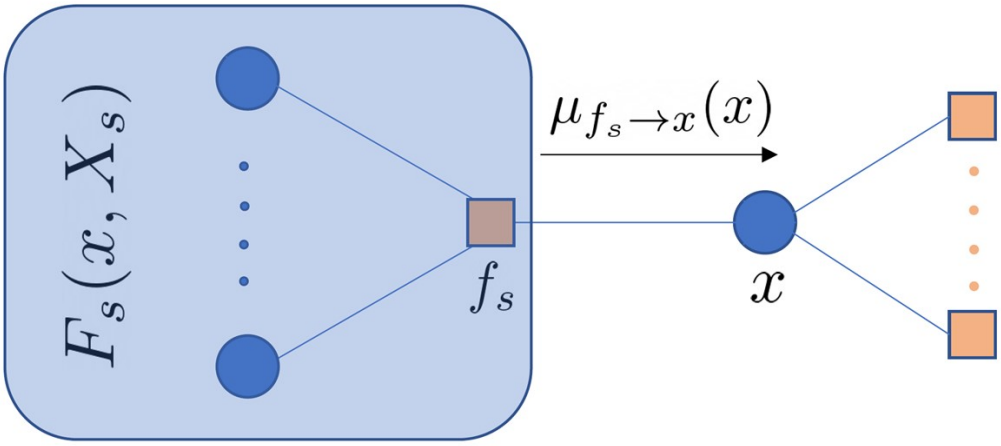


Figure 3.5: Factor to variable message (Andrew J. Davison) [14]

$$p(x) = \prod_{s \in n(x)} F_s(x, X_s) \tag{3.3}$$

In equation 3.3:

1. $n(x)$ is the set of factor nodes that are neighbours of x
2. F_s is the product of all factors in the group associated with f_s
3. X_s is the vector of all variables in the subtree connected to x via f_s

Combing the previous equation it's possible to obtain:

$$p(x) = \sum_{X \setminus x} \left[\prod_{s \in n(x)} F_s(x, X_s) \right] \tag{3.4}$$

To understand this process better, let us break it down step by step. Each term $F_s(x, X_s)$ represents a complex function that depends on the variable x and many other related variables from its part of the tree structure. In the equation, the first thing we do is multiply all these F_s terms together, combining them into one significant function that includes all the variables from the entire tree. After that, the sum of all the variables except for the one we are interested in is computed. This produces a function that only depends on x.

$$p(x) = \prod_{s \in n(x)} \left[\sum_{X_s} F_s(x, X_s) \right] \tag{3.5}$$

However, the process is a bit different if the sum and product are reordered as in the equation 3.5. Each F_s function from a branch is taken and sums over all the extra variables immediately, producing a simplified function that only depends on x. Once there's only a simple function for each branch, we multiply them together to get the final result, which is the marginal distribution of x. This approach can be better understood using "message passing," where each part of the tree sends information about x to combine it all together.

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s) \tag{3.6}$$

The term $\mu_{f_s \to x}(x)$ in equation 3.6 is a message sent from a specific graph part to the variable x. This message is represented as a probability distribution concerning$x$ alone. It is determined by considering all the information from that particular part of the tree, indicating what the probability of x is estimated to be by that section of the factor graph. When similar messages are received by x from all the connected parts of the tree, they can be combined. The final probability distribution of x is then computed by multiplying these messages together, es shown in equation 3.7 effectively integrating the information gathered from the various sections of the tree.

$$p(x) = \prod_{s \in n(x)} \mu_{f_s \to x}(x) \tag{3.7}$$

To go further into one of the branches of the tree, to compute the contribution of that sub-portion of variables of the tree, the product of factor $F_s(x, X_s)$ can be brake down as shown in equation 3.12:

$$F_s(x, X_s) = f_s(x, x_1, \cdots, x_M) \prod_{m \in n(f_s)} G_m(x_m, X_{S_m}) \tag{3.8}$$

Referring to Figure **??**, f_s, the factor which connects x to this branch, is a function of x as well as M other neighbouring variables $x_m \in x_1, \ldots, x_M$. Each of these variables connects to a sub-branch containing a product of factors G_m, a function of variable x_m and other variables $X_s m$. Substituting it into the equation 3.9:

$$\mu_{f_s \to x}(x) = \sum_{x_1, \cdots, x_M} f_s(x, x_1, \cdots, x_M) \prod_{m \in n(x)} \left[\sum_{X_{s1}, \cdots, X_{S_M}} G_m(x_m, X_{s_m}) \right] \tag{3.9}$$

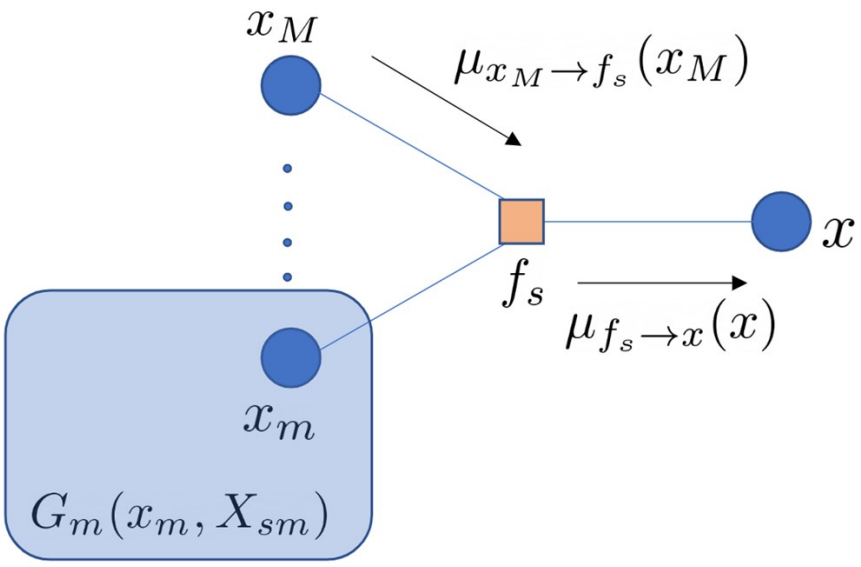


Figure 3.6: Factor to variable message (Andrew J. Davison) [14]

where leveraging the fact that $X_s = (x_1, \ldots, x_M, X_{S1}, \ldots, X_{SM})$ to separate out the sum. Is it possible to define the second type of message, this time from variable to factor:

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{s_m}} G_m(x_m, X_{s_m}) \tag{3.10}$$

Now substitute to get:

$$\mu_{f_s \to x}(x) = \sum_{x_1, \cdots, x_M} f_s(x, x_1, \cdots, x_M) \prod_{m \in n(x)} \mu_{x_m \to f_s}(x_m) \tag{3.11}$$

Consider Figure 3.7 , which now centers on x_m, one of the variable neighbors of f_s, which connects f_s to the product of factors $G_m(x_m, X_{s_m})$. We break down this product as follows:

$$G_m(x_m, X_{s_m}) = \prod_{l \in n(x_n) \setminus f_s} F_l(x_m, X_{ml}) \tag{3.12}$$

It is possible to notice that the total product factorizes into terms $F_l(x_m, X_{ml})$, each of which is the product of the set of factors from the whole graph, which connects to x_m via factor f_l. To boil down the set of all variables connected to f_s via x_m, X_{s_m} is broken down into subsets X_{ml}, which connect to x_m via factor f_l. If we further, substitute, it's possible to create the equation, that describes a message

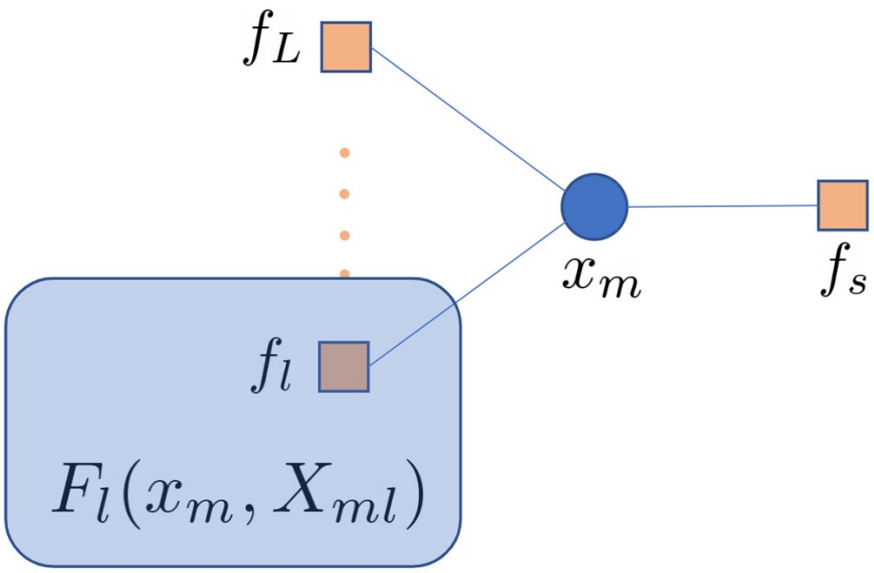


Figure 3.7: Contribution of factors (Andrew J. Davison) [14]

from factor to a variable.

$$\mu_{x_m \to f_s}(x_m) = \prod_{m \in n(x)} \mu_{f_l \to x_m}(x_m) \tag{3.13}$$

Once all the formals are retrieve in order, the algorithm iteratively apply the formula until convergence; each iteration consist of three phase:

1. Belief Update: The variable node beliefs are updated by taking a product of the incoming messages from all adjacent factors, each of which represents that factor's belief in the receiving node's variables.

2. Factor-to-variable message: To send a message to an adjacent variable node, a factor aggregates messages from adjacent variable nodes and marginalizes over all the other nodes' variables to produce a message expressing the factor's belief over the receiving node's variables.

3. Variable-to-factor message: A variable-to-factor message tells the factor what the belief of the variable would be if the receiving factor node did not exist. This is computed by taking the product of the messages the variable node has received from all other factor nodes.

BP can efficiently determine the marginal distributions of every variable in a tree graph through a single forward and backward message passing sweep. BP was

originally developed for graphs that are trees, and the updates were designed such that the beliefs converge to the exact marginals after one sweep of messages from a root node to the leaf nodes and back. For models with arbitrary conditional independence structure, including cycles or "loops", loopy BP iteratively applies the same message passing rules to all nodes. The simplest variant of loopy BP sends messages from all nodes at every iteration in a synchronous fashion.

As BP was initially developed for trees, its application to loopy graphs was at first empirical. Theoretical grounds for applying the same update rules to loopy graphs were later developed that explain loopy BP as an approximate variational inference method in which inference is cast as an optimization problem. Instead of directly minimizing the factor energies, loopy BP minimizes the Kullback–Leibler divergence between the posterior and a variational distribution, which we use as a proxy for the marginals after optimization. Loopy BP can be derived via constrained minimization of an approximation of the Kullback–Leibler divergence known as the Bethe free energy. As the Bethe free energy is non-convex, loopy BP is not guaranteed to converge, and even when it does, it may converge to the wrong marginals. Empirically, however, BP generally converges to the true marginals, although for very loopy graphs, it can fail.

We will now focus on Gaussian belief propagation, a specific type of continuous belief propagation for Gaussian models.

3.2.2 Gaussian Belief Propagation

GBP is particularly useful in scenarios where the relationships between variables and factors are linear and all factors follow Gaussian distributions. This makes GBP an effective tool for estimating the most likely values of continuous variables in systems like robotics. Upon convergence, GBP is known to compute exact marginal means, aligning with the results of a centralized maximum a posteriori solver.
However, it is important to note that the variances may be overestimated in cases involving loopy graphs, reflecting an overconfidence in the estimates.
While GBP does not guarantee convergence in every scenario, certain conditions and techniques can improve its reliability. For instance, tree-like structures are more likely to converge, and methods such as message damping can stabilize the process in more complex graphs.

To implement Gaussian Belief Propagation effectively, several key elements must be defined: the state representation, factor definitions, factor linearization strategies, the message-passing process, and robust factors to manage outliers

and unexpected data variations. When these components are carefully designed, GBP becomes a powerful approach for real-time inference and decision-making in robotics and other dynamic systems.

State Reppresentation

In GBP, the uncertainty in state variables is represented using Gaussian distributions. For a particular variable node m, the Gaussian distribution is typically expressed as in equation 3.14:

$$p_m(x_m) = Ke^{-\frac{1}{2}\left[(x_m-\mu_m)^T\Lambda'_m(x_m-\mu_m)\right]} \tag{3.14}$$

Where mu_m is the mean of the distribution, and Λ'_m is the precision matrix (the inverse of the covariance matrix). Alternatively, the Gaussian distribution can be represented in a different form reported in equation 3.15:

$$p_m(x_m) = K_2e^{-\frac{1}{2}x_m^T\Lambda_m x_m+\eta_m^T x_m} \tag{3.15}$$

In this form, η_m is the information vector related to the mean vector by $\eta_m = \Lambda_m\mu_m$. We use the information form because it allows us to handle cases where the Gaussian distribution is rank-deficient, meaning that the covariance form cannot capture directions of uncertainty.
The information form is particularly convenient in GBP because combining multiple distributions is straightforward: the information vectors and precision matrices can be added together. This property significantly simplifies the inference process, making it more efficient for robotics applications such as real-time state estimation.

Factor definition

In state estimation for robotics, accounting for the uncertainties inherent in sensor measurements is essential. Gaussian factors provide a powerful method for modeling these uncertainties within a probabilistic framework. Suppose a robot has a sensor designed to observe a quantity that depends on its state variables. When tested, the sensor's measurements differ from the expected values in a manner described by a Gaussian distribution. The associated Gaussian factor can be defined as reported in the equation:

$$f_s(x_s) = Ke^{-\frac{1}{2}\left[(z_s-h_s(x_s))^\top\Lambda_s(z_s-h_s(x_s))\right]} \tag{3.16}$$

In this expression, z_s represents the actual measurement from the sensor, while $h_s(x_s)$ is a function that describes how the measurement depends on the state variables x_s. The precision matrix Λ_s is the inverse of the covariance matrix and indicates the confidence level of the measurement. The normalizing constant K

ensures that the distribution is appropriately scaled, although its exact value is typically not necessary for inference.

Gaussian factors extend beyond just sensor measurements; they can also be used to incorporate prior knowledge. For example, a smoothness prior may assume that the robot's state changes gradually, which can help filter out noise in the sensor data.

To define a Gaussian factor, three essential components are required:

1. $h_s(x_s)$: This is the function that describes the relationship between the measurement and the state variables.
2. z_s: This represents the observed value of the measurement.
3. λ_s: This is the precision matrix, which reflects the confidence in the measurement.

In summary, these components work together to create a Gaussian factor in the context of state estimation.

Factor Linearization

In many real-world scenarios, the relationship between state variables and measurements is non-linear. To efficiently apply Gaussian Belief Propagation in such cases, it is necessary to approximate these non-linear factors with linear ones, a process known as factor linearization. Linearization enables the use of linear algebra techniques, simplifying the inference process. To linearize a non-linear factor, we first need to approximate the measurement function $h_s(x_s)$ around a given state estimate, x_0. Using a first-order Taylor series expansion, the measurement function can be approximated as:

$$h_s(x_s) \approx h_s(x_0) + J_s(x_s - x_0) \tag{3.17}$$

In the equation 3.17, J_s is the Jacobian matrix, which represents the derivative of the measurement function concerning the state variables. This linear approximation allows us to express the non-linear factor as a Gaussian factor in information form. The Gaussian factor is then defined by the information vector η_s and the precision matrix Λ'_s, calculated as follows:

$$\eta_s = J_s^\top \Lambda_s \left(J_s x_0 + z_s - h_s(x_0)\right) \tag{3.18}$$

$$\Lambda'_s = J_s^\top \Lambda_s J_s \tag{3.19}$$

In this context, η_s summarizes the influence of the measurement on the state variables, while Λ'_s reflects the confidence in the linearized measurement. These components enable the efficient propagation of information in the factor graph, even when dealing with non-linear relationships. By approximating non-linear factors with linear ones, we simplify the inference process while maintaining a high level of accuracy. This technique is crucial for applying Gaussian Belief Propagation in complex robotic systems, where non-linear sensor models are standard.

Message Passing

In the context of BP, messages are consistently represented as probability distributions within the state space of the variable node that either transmits or receives them. In the case of GBP, each message is expressed in terms of an information vector and a precision matrix, both defined within the corresponding state space.

A variable node x_m is typically connected to multiple factors. During a standard message-passing step, the node receives incoming messages from all connected factors except one and must produce an outgoing message to transmit to the remaining factor. All messages exchanged in this process are defined within the state space of the variable node x_m. The outgoing message is computed by combining all incoming messages through multiplication.

Each incoming message, denoted as $\mu_{f_l \to x_m}(x_m)$, is characterized by an information vector η_{ml} and a precision matrix Λ_{ml}. The information vector and precision matrix of the outgoing message represented as $\mu_{x_m \to f_s}(x_m)$, are derived by performing a summation operation as shown in equation 3.20:

$$\eta_{ms} = \sum_{l \in \mathcal{N}(x_m) \setminus f_s} \eta_{ml} \tag{3.20}$$

$$\Lambda_{ms} = \sum_{l \in \mathcal{N}(x_m) \setminus f_s} \Lambda_{ml} \tag{3.21}$$

When several Gaussian expressions are multiplied, their exponents are summed together. In GBP, messages form factor node process information from several connected variable nodes and sending a message to a specific target variable node. Each incoming message carries two pieces of information: a vector of values (η) and a precision matrix (Λ). These messages are combined by multiplying them together. The result is then adjusted using the factor potential, which describes how all the connected variables, including the target variable, are related.

Because the factor potential often involves complex, non-linear relationships, it is simplified using linearization. This means that the factor potential is approximated

as a straight-line relationship based on an initial guess of the variable states, labeled as x_0. This simplification creates a new set of values: an updated information vector (η_s) and a precision matrix (Λ'_s). These values are calculated once in each process step or less often if the relationships are already linear.
Once the incoming messages and the simplified factor potential are combined, the next step is to focus only on the target variable by marginalizing the other variables. This involves rearranging the values to place the target variable at the top of the list. For instance, if the target variable is m_3, the rearranged values look as follows:

$$\eta_{CRs} = \begin{pmatrix} \eta_{Csm3} \\ \eta_{Csm1} \\ \eta_{Csm2} \end{pmatrix}, \quad \Lambda'_{CRs} = \begin{bmatrix} \Lambda'_{Csm3m3} & \Lambda'_{Csm3m1} & \Lambda'_{Csm3m2} \\ \Lambda'_{Csm1m3} & \Lambda'_{Csm1m1} & \Lambda'_{Csm1m2} \\ \Lambda'_{Csm2m3} & \Lambda'_{Csm2m1} & \Lambda'_{Csm2m2} \end{bmatrix} \tag{3.22}$$

Using the reordered form show in equation, the marginalized distribution over the output variable m_3 is obtained as:

$$\eta_{M\alpha} = \eta_\alpha - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\eta_\beta, \tag{3.23}$$

$$\Lambda_{M\alpha} = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\beta\alpha} \tag{3.24}$$

This final marginalized distribution forms the outgoing message $\mu_{f_s \to x_{m3}}$, sent to the variable node m_3.

3.2.3 Robust Factors

In fields like computer vision and robotics, it's typical for sensors, especially outward-facing ones, to give measurements that are not perfectly Gaussian. Usually, the sensor's measurements follow a pattern close to a Gaussian distribution when the sensor works properly. These measurements are usually close to the actual values, with only slight differences caused by rounding or minor errors. However, sometimes the sensor might give wrong measurements or "outliers," far from the actual values. For example, a camera might misidentify the position of an image feature, leading to measurements that are far off from the actual location.
Looking at the distribution of these sensor measurements, they usually have a Gaussian pattern mainly in the center but with "heavy tails" on the edges, meaning there are more extreme outlier values than a perfect Gaussian distribution would show. To deal with these outliers in estimation and optimization, a special class of functions called M-Estimators are designed to handle such situations.
In GBP, each message is assumed to originate from a Gaussian distribution characterized by an information vector and a precision matrix. When outlier measurements are identified, the representation is adjust using, for example, the Huber function, which helps mitigate the impact of extreme values. The Mahalanobis distance is

used to find outliers by measuring a value's distance from the expected mean. If this distance exceeds a certain threshold (set by a factor σN), the "robust zone" is entered. The precision matrix and information vector are scaled in this zone using a factor k_R to handle the outliers better. The adjusted Mahalanobis distance is:

$$M_{sR} = \sqrt{2N\sigma M_s - N\sigma^2} \tag{3.25}$$

The scaling factor, in the case of a Huber loss function, k_R is calculated as:

$$k_R = \frac{M_{sR}^2}{M_s^2} = \frac{2N\sigma M_s - N\sigma^2}{M_s^2} \tag{3.26}$$

This adjustment reduces the influence of the outlier for this message pass, ensuring that the overall estimation remains robust.

Chapter 4

Implementation

This thesis aims to investigate the capability of factor graphs to model the complexities of multi-robot systems and examine how GBP enables efficient inference within such models. This research specifically focuses on applying these techniques to the problem of multi-robot localization, achieved by fusing odometry and UWB measurements in a decentralized robotic swarm.
The localization technique will be implemented in a decentralized swarm of robots, each equipped with a UWB anchor and operating in conjunction with a set of external fixed UWB anchors at known positions. While the swarm of robots shares a common clock, they are not synchronized, meaning that each robot will perform relocalization at distinct time instants.
In order to meet the objectives of this project, three essential tasks need to be completed:

1. **Design of the Factor Graph Model:** Develop a robust framework that accurately represents the relationships between odometry, UWB measurements, and the robot's position.

2. **Implementation of the GBP Algorithm:** Create an efficient implementation of the algorithm capable of performing inference on the factor graph, handling noise, and optimizing performance.

3. **Asynchronous Data Management:** Since the robots share a common clock but are not synchronized, direct message passing between variables is not feasible. Consequently, data interpolation is required to estimate the position of a robot at the time an intra-robot measurement is performed.

4.1 Design of the Factor Graph Model

This section explains the process of constructing the factor graph. The primary purpose of the factor graph is to integrate information from three key sources: prior knowledge, odometry, and UWB measurements. These data sources provide complementary information, enabling robust localization in complex multi-robot environments.

GBP will serve as the optimization algorithm to estimate the robot's position. To ensure the effectiveness of GBP, it is essential to design the factor graph in a manner that utilizes its properties. A well-constructed factor graph will promote the rapid convergence of GBP to a precise solution.

4.1.1 Prior factor

The design of a factor graph that updates a variable through prior knowledge is straight for work; the structure is shown in the Figure 4.1.

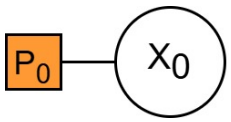


Figure 4.1: Factor graph variable with prior

The state variables X_n of the factor graph are represented by an mean vector μ_n and a precision matrix Λ_n, each of dimension two, in order to localize the robot position on a plane, as shown in the following Equation 4.1:

$$\mu_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix}; \Lambda_n = \begin{bmatrix} xx_n & xy_n \\ yx_n & yy_n \end{bmatrix} \tag{4.1}$$

A prior refers to knowledge about a specific set of factor graph variables. In this scenario, it belongs to the starting position of the robot. Therefore, the prior factor is connected to the first variable node, X_0, which represents the initial state of the robot.
The mathematical equation defining the prior factor is straightforward, as shown in 4.2:

$$h_s(X_s) = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{4.2}$$

The measurement and the precision matrix of the measurement are shown in

Equation 4.3:

$$z_s = \begin{bmatrix} z_x \\ z_y \end{bmatrix}; \Lambda_s = \begin{bmatrix} z_{xx} & z_{xy} \\ z_{yx} & z_{yy} \end{bmatrix} \tag{4.3}$$

In Equation 4.3, z_x and z_y represent the x and y coordinates of the robot's initial position, respectively. The term Λ_s denotes the inverse of the variance matrix, which quantifies the accuracy of the robot's initial position estimate.

4.1.2 Odometry information

The odometry data are used to relate two sequential state variables, allowing the robot to reposition itself based solely on internal measurements. The structure of the graph is built upon the one shown in Figure 4.1, which is illustrated in Figure 4.2.

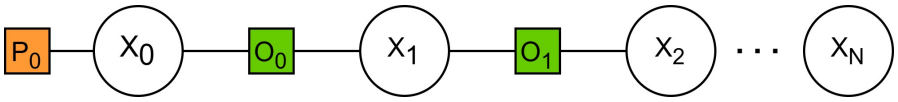


Figure 4.2: Factor graph with odometry data

The odometry factor O_n that interconnects two sequential robot poses is characterized by the mathematical equation:

$$h_s(X_s) = \begin{bmatrix} x_n - x_{n-1} \\ y_n - y_{n-1} \end{bmatrix} \tag{4.4}$$

The measurement and its corresponding precision matrix are presented in Equation 4.3. Typically, the term Z_s is calculated using the following equation in the literature:

$$Z_s = \begin{bmatrix} \delta_{trasl} \cos(\delta_{rot1} + \theta) \\ \delta_{trasl} \sin(\delta_{rot1} + \theta) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix} \tag{4.5}$$

In this equation, δ_{transl}, δ_{rot1}, and δ_{rot2} represent data from the odometry sensor, while θ is the robot's most recent known rotation.
There is a limitation in the scenario analyzed in this thesis, which relies on both UWB and odometry data for robot localization. The rotation errors accumulated by the odometry sensor cannot be corrected using UWB measurements because UWB cannot measure the robot's rotation. As a result, the θ parameter quickly becomes highly inaccurate due to growing errors. These errors make the odometry data unreliable for use in pose estimation.
Although odometry alone accumulates errors over time and becomes unusable, combining it with UWB data can help filter out incorrect information from the

glsUWB sensor. To achieve this, even without the ability to correct the robot's rotational estimation, the odometry data in this thesis is used as a regularization term between two relocation instances.
This is achieved by setting the value of Z_s to zero, which regularizes the robot's trajectory. Meanwhile, the odometry data is utilized to compute the Λ_s matrix, as illustrated in the following equations:

$$\Lambda_s = \begin{bmatrix} 2\delta_{trasl}{}^2 & 0 \\ 0 & 2\delta_{trasl}{}^2 \end{bmatrix}^{-1} \tag{4.6}$$

As shown in Equation 4.6, the diagonal elements represent the variance of the regularization term. This variance is based on the distance between the robot's previous position and its new position. By doing this, the regularization term adjusts to the measurements provided by the odometry. This allows it to adapt effectively in situations where the robot relocates at irregular intervals or moves at varying speeds.

4.1.3 UWB data

UWB data determines the robot's position within a global reference frame. To construct the factor graph, each variable representing the robot's pose must be connected to another variable that represents either the UWB position or another robot's already known position. The locations of the UWB anchors are predetermined, while the robot's position is shared through ROS2 and updated right before running the inference algorithm. The final factor graph, based on the one shown in Figure 4.2, is presented in Figure 1.12.

In Figure 4.3, each variable node A_n represents either a UWB anchor or the robot's position. Each factor U_n corresponds to a UWB measurement, while the prior factor P_{A_n} provides known information about the positions of UWB anchors or other robots.

However, due to the property of the GBP algorithm, it loses its ability to converge to a solution if the graph contains loops. At the same time, increasing the number of loops is often necessary to improve the accuracy of robot localization in this type of setup. A different graph structure is used to address this issue. This new structure effectively models the localization problem while preserving the convergence capability of GBP, as shown in Figure 4.4.

The factor graph shown in Figure 4.4 represents a scenario with three UWB anchors and three robots. Each factor Ua_t corresponds to the UWB measurement of agent a during relocation t. Each variable Aa_t represents the position of agent a at relocation t. Lastly, each prior factor PAa_t provides information about the

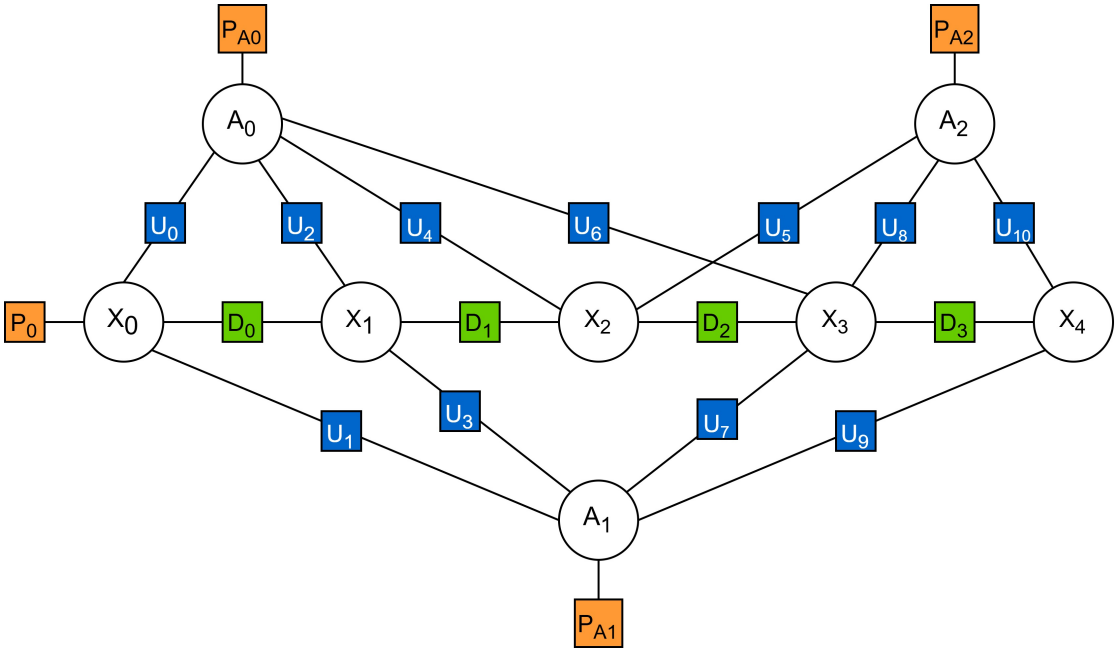


Figure 4.3: Loopy factor graph with UWB and Odometry

position of the anchor or robot a at relocation t.

Regarding the factor definitions, the prior factors and the odometry factor are defined in Equations 4.2, 4.3, 4.4 and 4.6, while the UWB factor is defined in Equation 1.5:

$$h_s(X_s) = \sqrt{(Aa_{tx} - X_{tx})^2 + (Aa_{ty} - X_{ty})^2 + dz^2} \tag{4.7}$$

$$Z_s = UWB_{range} \tag{4.8}$$

$$\Lambda_s = \lambda_{uwb} \tag{4.9}$$

Equation 4.7 presents the mathematical model that establishes the relationship between the position of the robot and that of an external UWB anchor or another robot. The UWB sensor provides distance measurements, and thus, the relationship between the two variables in the factor graph is expressed using the Euclidean distance formula. In this equation, $Aa_{tx/y}$ denotes the x and y coordinates of the measured agent, which may be either a fixed UWB anchor or another robot while $X_{tx/y}$ are the robot x and y coordinates. The term dz accounts for the height difference between the robot and the UWB anchor. The measurement obtained from the UWB sensor represents the range (distance), while the corresponding

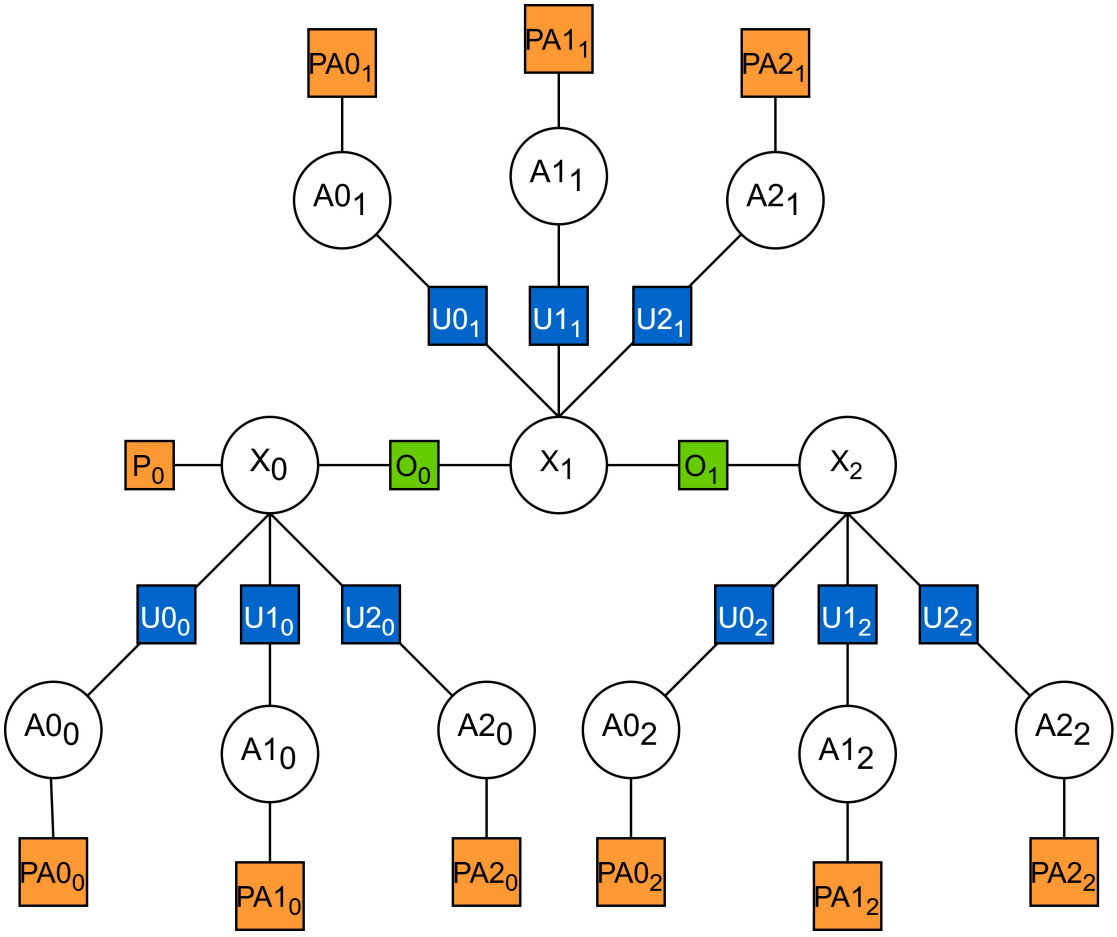


Figure 4.4: Non-Loopy factor graph with UWB and Odometry

precision matrix, indicative of the measurement's reliability, is determined as the inverse of the variance of the UWB measurement.

4.2 Implementation of the GBP algorithm

This section is dedicated to the implementation of the GBP algorithm. For its successful implementation, several critical requirements must be addressed:

1. Scalability: The GBP algorithm must exhibit scalability, ensuring that the computational time is not significantly influenced by the number of robots or ultra-wideband (UWB) antennas providing data.

2. Dynamic Agent Management: The algorithm must be capable of accommodating a variable number of agents, each undergoing relocalization at distinct

time intervals.

3. Noise Robustness: The GBP algorithm must demonstrate the ability to process noisy data without undermining the accuracy of the results.

4. Efficient Convergence: It is imperative that the algorithm converges to a solution within a reasonable timeframe to support practical applications.

The requirement for efficient convergence is inherently fulfilled by the structure of the factor graph presented in Figure 4.4. Due to the absence of loops in the graph, the GBP algorithm can achieve convergence to the exact solution through a systematic sequence of message passing, progressing from the root to the leaf nodes and then returning to the root. However, the non-linear characteristics of the UWB factors necessitate additional iterations. A single sequence of message passing is insufficient; instead, multiple iterations are required. Each iteration involves linearizing the non-linear factors and repropagating the messages to refine the solution.

To achieve the stated objective, it is essential to utilize the available computational resources efficiently. This is accomplished by employing the PyTorch library, which facilitates the creation of tensors capable of accurately representing the structure of the factor graph. Furthermore, PyTorch's built-in functions are leveraged to perform high-performance computations on these tensors, ensuring both accuracy and computational efficiency.

4.2.1 Messages generation

The objective of the GBP algorithm is to estimate the unknown variables within the factor graph, which, in this context, correspond to the poses of the robots. As outlined in the theoretical background chapter, the algorithm operates by iteratively executing three primary steps: the generation of variable-to-factor messages, the generation of factor-to-variable messages, and the update of beliefs.

The messages exchanged between factors and variables are represented as Gaussian distributions in information form. These messages consist of a precision matrix, denoted as λ, and an information vector, denoted as η. This representation facilitates straightforward computation of variable-to-factor messages and belief propagation, as the process is independent of the specific type of variable or factor.

To perform the belief update, it is necessary to aggregate all the messages received from the factors connected to a given variable. Specifically, all associated information vectors and precision matrices are summed to refine the estimate of the variable. Similarly, the computation of variable to factor messages requires

summing all the messages from factors located on the branches of the factor graph that are connected to the variable, excluding the branch along which the message is being sent.

Generating factor-to-variable messages represents the most intricate step in the GBP process, consuming most of the computational resources. This step involves three distinct and essential operations, regardless of the specific type of factor:

1. Factor linearization: The non-linear factor is linearized around a reference point.
2. Partitioning, conditioning, and reordering of the factor's information vector and precision matrix: The components of the factor are systematically organized, conditioned, and reordered to facilitate easy computation of variable marginalization.
3. Marginalization of the output variable: The output variable is marginalized in order to produce the required message for propagation.

The factor linearization can be obtained through Equation 3.18, where J_s represents the Jacobian matrix of the mathematical equation that defines the factor. For the odometry factor shown in the Equation 4.4, and UWB factors shown in Equation 4.7 , the corresponding Jacobian matrices are provided in the following Equations:

$$J_{Odometry}(x_{n-1}, x_n, y_{n-1}, y_n) = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \tag{4.10}$$

$$J_{UWB}(X_{tx}, X_{ty}, Aa_{tx}, Aa_{ty}) = \begin{bmatrix} \frac{-(Aa_{tx}-X_{tx})}{\sqrt{(Aa_{tx}-X_{tx})^2+(Aa_{ty}-X_{ty})^2+dz^2}} \\ \frac{-(Aa_{ty}-X_{ty})}{\sqrt{(Aa_{tx}-X_{tx})^2+(Aa_{ty}-X_{ty})^2+dz^2}} \\ \frac{(Aa_{tx}-X_{tx})}{\sqrt{(Aa_{tx}-X_{tx})^2+(Aa_{ty}-X_{ty})^2+dz^2}} \\ \frac{(Aa_{ty}-X_{ty})}{\sqrt{(Aa_{tx}-X_{tx})^2+(Aa_{ty}-X_{ty})^2+dz^2}} \end{bmatrix}^T \tag{4.11}$$

Equations 4.10 and 4.11 show that the Jacobian of the odometry is linear, whereas the Jacobian of the UWB is nonlinear. Consequently, the UWB Jacobian will require re-linearization with each iteration around a new estimation point.

The subsequent steps, applicable identically to every type of factor, involve partitioning, conditioning, and reordering the information vector and precision matrix to achieve a matrix structure consistent with the format presented in the equation. Finally, the distribution must be marginalized over the output variable to generate the output message, as specified in Equation 3.23.

4.2.2 Robust factor

GBP is inherently designed to operate with Gaussian distributions. However, to effectively manage non-Gaussian data distributions, robust factors utilizing M-estimators are employed to increase performance in noisy scenarios. Various M-Estimators can be utilized; some examples are presented in the following figure.

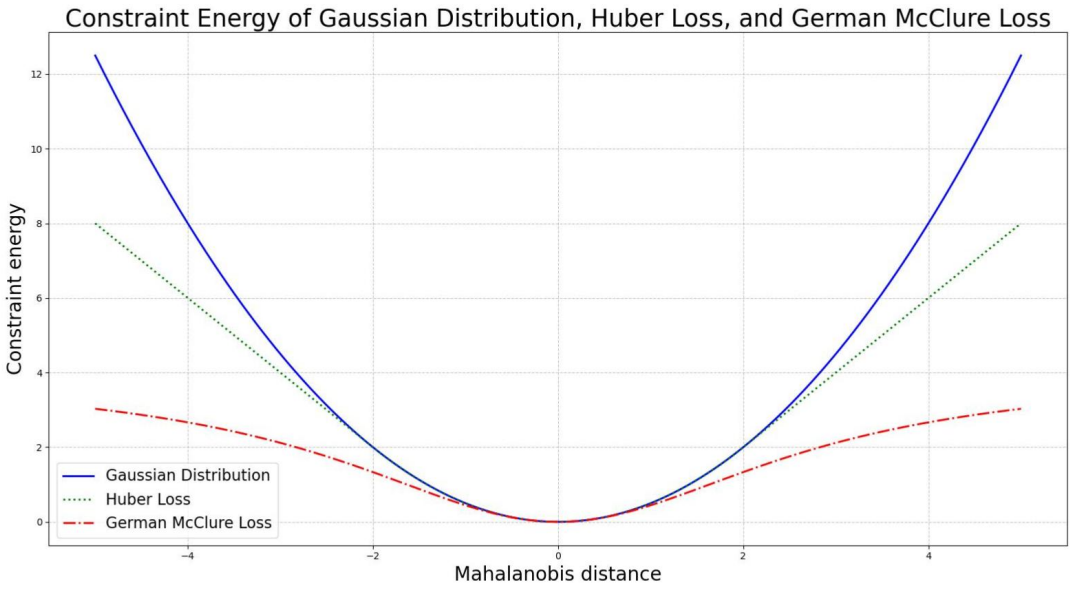


Figure 4.5: Constrain energy for the Gaussian Distribution, Huber Loss, and German McClure Loss

The Figure 4.5 illustrates how the energy of the constraint varies as a function of the Mahalanobis distance for each M-estimator function. In this thesis, the Geman-McClure loss function is selected due to its ability to significantly reduce the energy of a constraint, compared to the Gaussian energy, as the Mahalanobis distance increases. This property makes it particularly effective in diminishing the impact of erroneous measurements.
To apply the Geman-McClure loss function while preserving the Gaussian nature of the constraint, the Gaussian energy must be appropriately scaled. This process involves multiplying the information vector η_s and the precision matrix μ_s by a scaling coefficient k_R, which must be recalculated for each message generated. The energy of the Gaussian constraint is expressed as $\frac{1}{2}M_s^2$, while the energy of the Geman-McClure loss function is represented by the expression $\frac{aM_s^2}{b+M_s^2}$. The scaling coefficient is obtained by calculating the Mahalanobis distance at which the standard quadratic energy is equal to the energy of the Geman-McClure function.

Specifically, it is necessary to compute M_{sR}, the Mahalanobis distance at which this equivalence occurs.

$$\frac{1}{2}M_{sR}^2 = \frac{aM_s^2}{b + M_s^2} \tag{4.12}$$

Rearranging is obtained:

$$M_{sR} = \sqrt{\frac{2aM_s^2}{b + M_s^2}} \tag{4.13}$$

And therefore:

$$k_R = \frac{M_{sR}^2}{{M_s}^2} = \frac{2a}{b + M_s^2} \tag{4.14}$$

K_r is the scaling coefficient that depends on two other coefficients, a and b. These coefficients can be easily adjusted by performing a parameter sweep on both and compiling a table to identify the optimal values. Alternatively, they can be set based on system knowledge. In this thesis, the values are set to $a = 2$ and $b = 4$ to penalize all UWB measurements that provide an estimated robot position that is at least one standard deviation away from the determined position.

The robust factor can be further developed by integrating knowledge about the physical principles underlying UWB measurements. UWB sensors determine distances based on the time of flight of electromagnetic signals between two antennas. Therefore, the measured distance cannot be less than the distance between the antennas, as this would imply a signal speed exceeding the speed of light. To include this physical constraint in the factor model, both the Mahalanobis distance and the difference between the UWB-measured distance and the estimated distance are considered.

$$dist_{diff} = UWB_{range} - \sqrt{(Aa_{tx} - X_{tx})^2 + (Aa_{ty} - X_{ty})^2 + dz^2} \tag{4.15}$$

Equation 4.15 calculates the discrepancy between the UWB measurement and the estimated distance between two UWB antennas. If this difference is negative, the robot's estimated position corresponds to an unattainable distance relative to the UWB antenna. Accordingly, the robot's function is modeled as shown in Equation 4.17.

$$M_{uwb} = \text{sign}(dist_{diff}) \cdot M_S \tag{4.16}$$

$$K_{uwb} = \begin{cases} c \cdot M_{uwb}^2 & \text{if } M_{uwb} < 0 \\ \frac{2 \cdot a}{b + M_{uwb}^2} & \text{if } M_{uwb} \geq 0 \end{cases} \tag{4.17}$$

Equation 4.17 defines a robust factor represented by a continuous function with a continuous first derivative. The coefficient c is assigned a sufficiently large value to ensure that the message scaled by this coefficient becomes the most influential. Specifically, c is set to be two orders of magnitude greater than the largest scaling factor in the corresponding message generation step, ensuring that the new estimated point adheres to the constraint. The shape of the modified robust factor is illustrated in the following Figure 4.6.

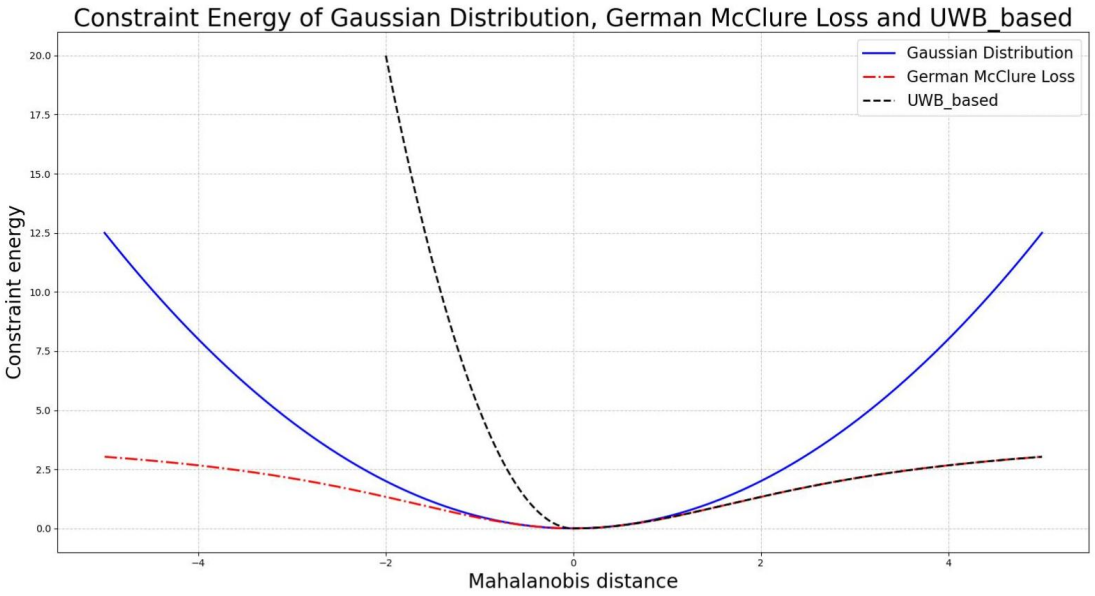


Figure 4.6: Constrain energy for the Gaussian Distribution, German McClure Loss and UWB based

4.2.3 Initialization and Sliding Windows

The initialization process is a critical step for the effective execution of the GBP algorithm and for achieving the desired outcomes. Two types of data must be initialized: the variable values and the messages.

Two distinct scenarios must be considered regarding the initialization of variables: the startup of the system and the addition of a new state to the graph. During the system startup, all variables in the factor graph are initialized with random mean values and large variances, typically on the order of several meters. Conversely, when a new state variable is added due to a new relocalization event, the mean value of the variable is initialized to match the previous state variable,

while its variance is doubled to account for increased uncertainty.

As for the initialization of messages, all messages are assigned an initial zero mean value and a variance of ten meters. This ensures that newly initialized messages do not contribute to the estimation process until they are explicitly updated during the factor-to-variable message generation step. This approach facilitates a smooth and unbiased integration of new data into the estimation framework.

This type of initialization allows for the management of a variable number of robots composing the swarm. It is sufficient to create a factor graph that models the interactions with the maximum number of robots it could potentially interact with. Then, we simply update the factor data of the robots that interact at a specific moment in time.

Sliding window techniques are employed to address the continuous expansion of the factor graph, which grows each time the robot relocalizes. As the graph's size increases, performing real-time inference across the entire graph quickly becomes computationally infeasible. To maintain computational efficiency, the number of past time steps included in the analysis is restricted, an approach commonly referred to as the sliding window method.
The choice of the window size, or the number of past time steps to consider, is typically influenced by the specific nature of the problem being modeled. While existing literature often suggests using fewer than ten past time steps, the optimal value can vary depending on the application. In this thesis, extensive testing revealed that a sliding window encompassing four past time steps provides the best balance between computational efficiency and performance.

4.3 Asynchronous Data Management

A critical requirement for effectively operating a real swarm of robots is the ability to manage asynchronous data efficiently. The GBP algorithm inherently addresses this challenge due to its local nature, enabling it to seamlessly integrate asynchronous data updates. New sensor measurements or state updates provided by individual robots can be incorporated without compromising the inference capabilities of the GBP algorithm.
In the context of a robotic swarm, the state variables within the sliding window of the factor graph are shared among all robots. However, the robots in the swarm are not synchronized, meaning that the timing of each robot's relocalization differs from the others. This asynchrony requires a method for data synchronization, involving interpolation of information from external sources. To achieve this, Gaussian

Process Regression (GPR) is employed, allowing for interpolation that accounts for both the mean values and the precision matrices of the data. A radial kernel is utilized to expand the variance between interpolation points, ensuring robust and realistic modeling of uncertainty.
An example of position interpolation is illustrated in Figure 4.7. The red dots represent the positions reported by a robot in the swarm, while the blue line shows the interpolated trajectory using GPR. The shaded gray region indicates the expanded variance between known points, as modeled by the radial kernel.

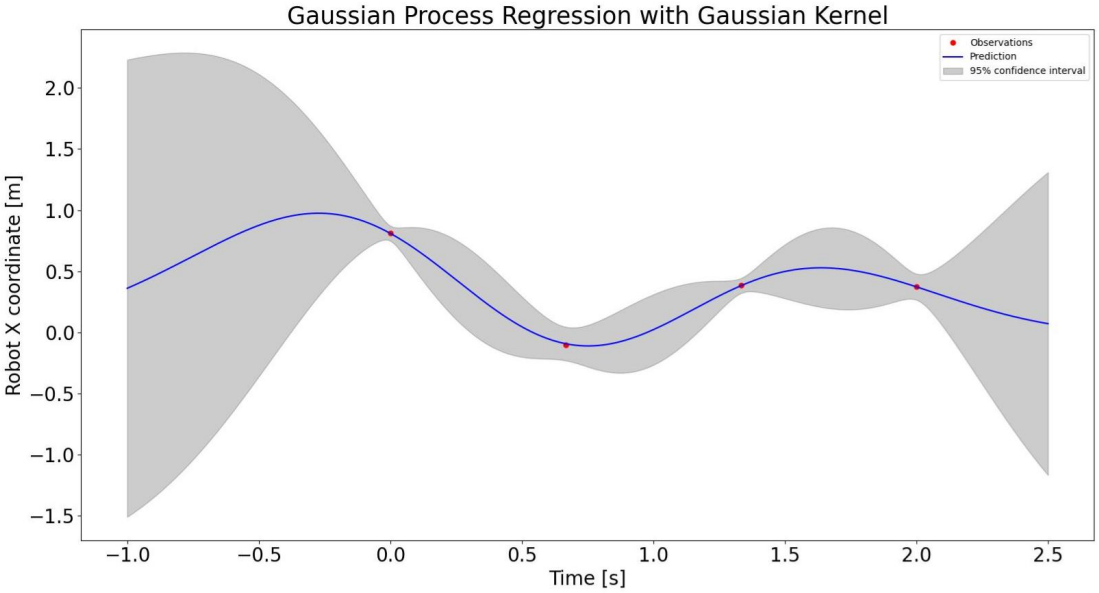


Figure 4.7: Example of Gaussian Process Regression used for interpolation

Chapter 5

Experimental Setup

A fundamental step of this work is the collection of real-world data to evaluate the capability of the proposed solution to merge wheel speed information and UWB measurements. To collect the necessary data, an experiment involving four TurtleBot3 robots was conducted in the laboratory of the Interdepartmental Center PIC4SeR. The purpose of this experiment was to collect data that mimics a real-world scenario in which a small swarm of robots shares an environment to carry out their operations. During the experiment, each robot was tasked with reaching a randomly generated goal position. To achieve this without colliding with other robots, a Dynamic Window Approach (DWA) local planner was employed by each robot. This created a scenario in which the swarm of four TurtleBot3 robots could navigate autonomously within the laboratory. The data were collected for almost an hour, with the robots moving at various velocities and accelerations.

5.1 Dataset acquisition

The laboratory experiments were conducted a the interdepartmental center PIC4SeR with four TurtleBot3 robots, shown in Figure 5.2.

Each robot was equipped with a DWM1001C Qorvo UWB module for intra-robot distance measurements, shown in Figure . To enable global localization, six additional UWB modules of the same type were strategically placed around the laboratory's perimeter. Each robot performed UWB measurements at a frequency of 2 Hz. However, measurement errors occasionally resulted in missing data points. Furthermore, the robots operated asynchronously, with each taking UWB measurements at distinct time instants.

To validate the performance of the proposed solution, a Vicon motion capture system, comprising eleven Bonita cameras, was employed to provide accurate

Figure 5.1: TurtleBot3 robot swarm

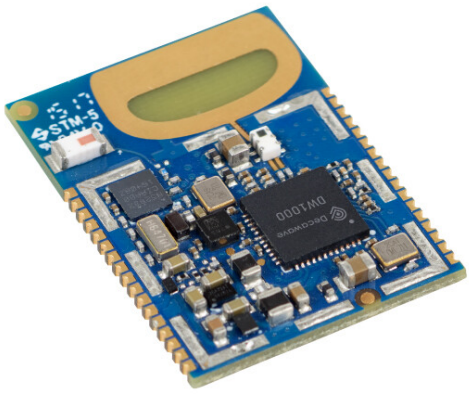


Figure 5.2: UWB module DWM1001C Qorvo

ground-truth localization. This system ensured millimeter-level precision in tracking the robots within the global reference frame, facilitating a rigorous evaluation of the solution's effectiveness.

The robot swarm operated in the laboratory for nearly an hour. After the

test, the performance of the UWB sensors was evaluated by comparing the measurements taken by the UWB antennas to data from the Vicon motion capture system. The measurements exhibited a significant error, the errors distribution of a representative ROS2 bag is shown in Figure 5.3, with a root mean square error (RMSE) of 33.58 cm.

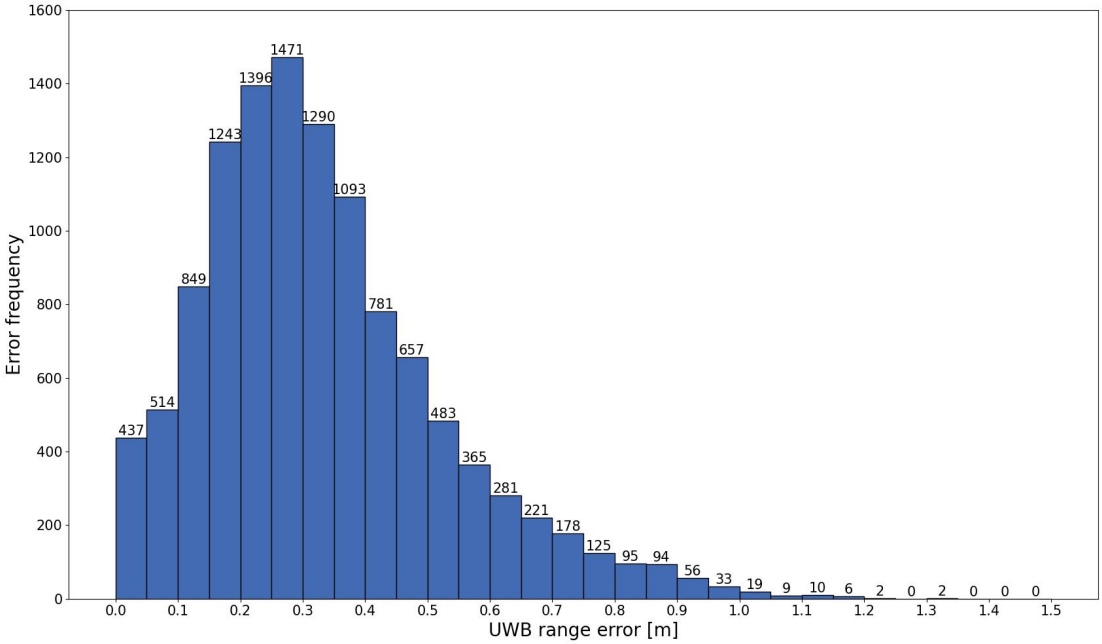


Figure 5.3: UWB range error distribution

Chapter 6

Result and Analysis

This chapter evaluates the proposed multi-robot localization framework based on GBP on factor graphs. The analysis encompasses both simulation-based scalability tests and real-world experiments, highlighting the algorithm's performance under varying conditions and constraints.

In the simulation environment, the focus is on assessing the scalability and robustness of the GBP algorithm. Key aspects evaluated include the algorithm's ability to converge as the noise levels in odometry and UWB measurements increase and its scalability as the size of the robotic swarm grows. These tests provide insight into the behavior of GBP in scenarios that closely mimic real-world challenges in multi-robot systems.

he real-world experiments are designed to validate the algorithm's performance in practical applications. Two distinct scenarios are evaluated. In the first scenario, all robots are able to receive information from all available sensors. In the second scenario, one robot in the swarm is unable to perform measurements using the UWB anchors positioned around the laboratory perimeter and must rely exclusively on intra-robot measurements for localization.

The results presented in this chapter aim to demonstrate the algorithm's effectiveness in addressing practical constraints of multi-robot systems, its adaptability to noise and limited sensor data, and the benefits of robust factor integration. By combining simulation and real-world evaluations.

6.1 Simulation

This section presents the simulation-based evaluation of the proposed solution. The results focus on two critical aspects: the scalability of the system as the number of robots in the swarm increases and the algorithm's convergence performance under progressively noisier sensor data.

Solution Scalability

To address real-world robotics challenges, it is imperative that the proposed solution demonstrates scalability when deployed on standard CPUs. Modern mobile robotics platforms typically feature limited computational resources due to size, weight, and power constraints. Consequently, algorithms used in multi-robot systems must be optimized to fully leverage the available computational capacity without overloading the system or compromising performance.

This necessity becomes even more critical when scaling to larger swarms, where the computational demands grow significantly due to the increased number of robots. Algorithms that fail to scale efficiently can lead to delayed inference, reduced responsiveness, and potential system failures in real-world applications.
This evaluation tests the proposed solution on an Intel i7-7700HQ CPU @2.80GHz.

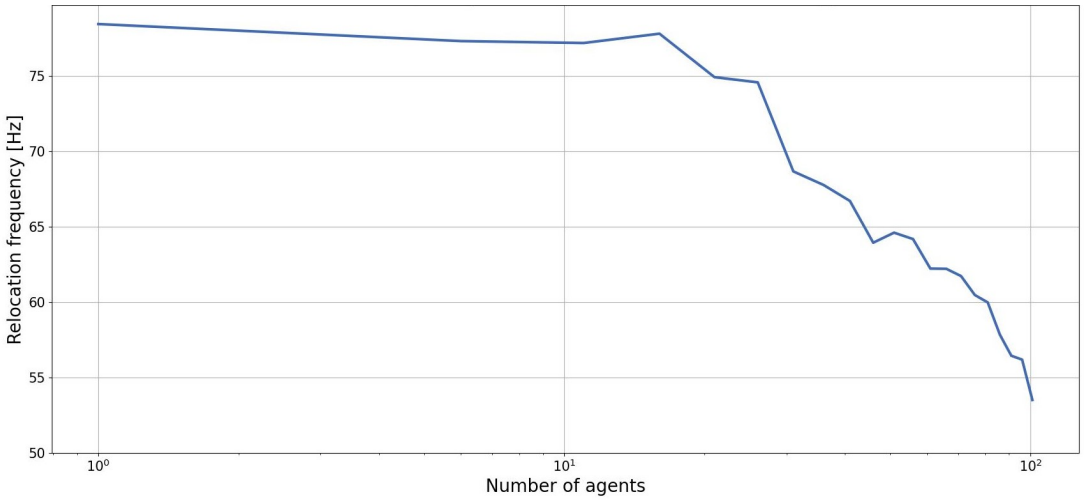


Figure 6.1: GBP inference frequency at the increasing of the swarm

The factor graph incorporate the information from three past time instant and the GBP iterate ten times to converge to the solution in each scenario. The test focuses on evaluating the system's ability to handle swarms of up to one hundred robots. This approach highlights the solution's practicality for deployment in large-scale,

resource-constrained robotics systems.

The results shown in Figure 6.1 demonstrate the scalability of the proposed solution as the swarm size increases from one to one hundred robots. Initially, the system maintains a relatively stable performance, with minimal variation in frequency as the swarm size grows from one to twenty robots. However, as the swarm size continues to increase beyond twenty robots, a gradual decline in frequency is observed, becoming more pronounced as the number of robots approaches one hundred. When the swarm size reaches its peak, the frequency drops to about 53.5 Hz.
Despite this decline, the system demonstrates a reasonable level of scalability, maintaining operational feasibility up to one hundred robots. These results validate the potential applicability of the proposed solution to real-world scenarios.

Convergence in noisy scenario

This subsection examines the impact of increasing sensor noise on the convergence behavior of the proposed multi-robot localization framework. To evaluate the robustness of the algorithm under adverse conditions, simulations were conducted with progressively larger noise levels applied to these sensors.The noise introduced in the simulations is modeled to reflect typical sources of error in real-world robotic systems.
The odometry data is perturbed by an unbiased Gaussian noise with a standard deviation ranging from 5 cm to 10 cm. The UWB measurements are affected by a positive uniform distribution with noise magnitudes ranging from 5 cm to 15 cm. In the simulation, the factor graph is applied using a sliding window of dimension four, with the algorithm iterating ten times during each relocalization. In each scenario, the robot relocalizes five hundred times, and the worst-case scenarios in terms of the number of iterations are showed in Figure 6.2.
Thanks to the non-loopy factor graph that describes the system, the algorithm consistently converges in very small amount of iterations, always less than yen, within the simulated scenarios.

6.2 Real-World Experiments

This chapter presents the real-world validation of the proposed multi-robot localization framework using a swarm of four TurtleBot3 robots. Each robot is equipped with a UWB antenna for intra-swarm measurements, additionally, six fixed UWB antenna are strategically deployed to provide global reference frame localization for the robots.

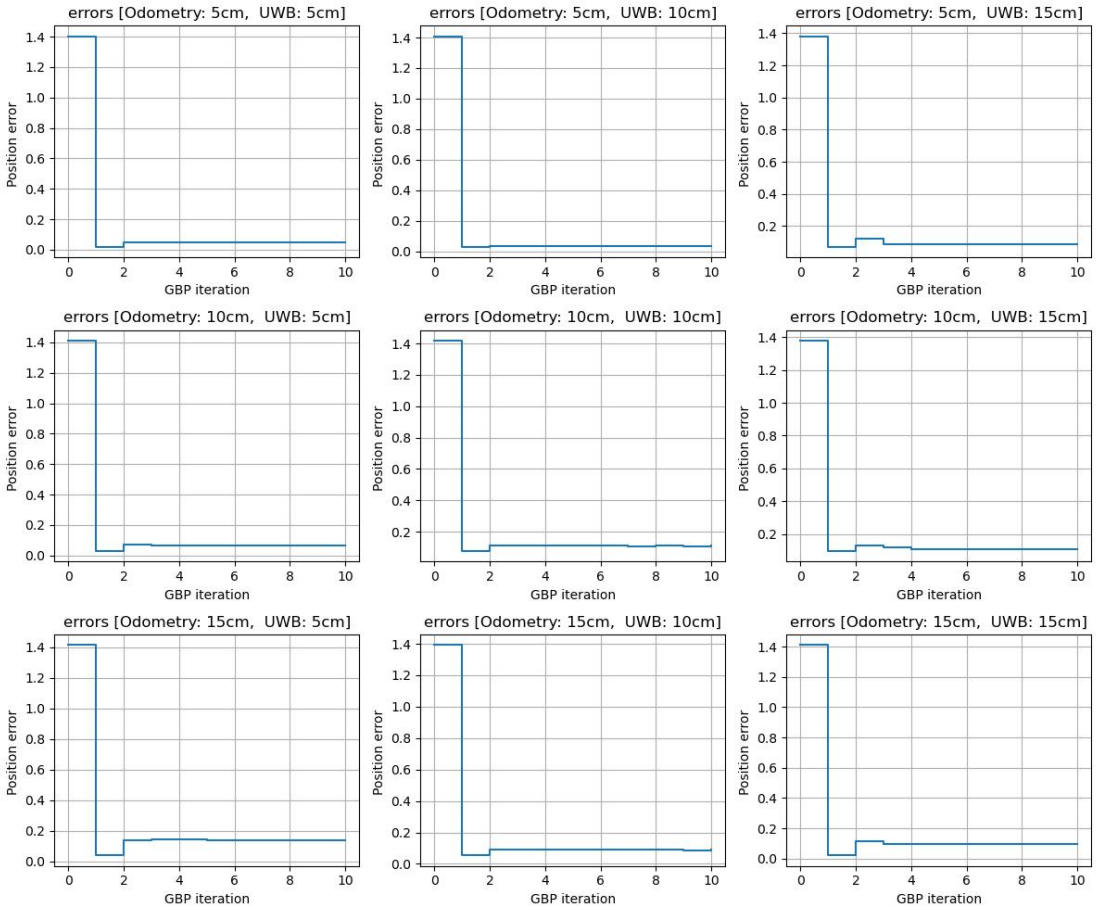


Figure 6.2: Convergence to a solution in sensors affected by noise

The experiments are designed to evaluate the practical performance of the localization framework and its ability to handle real-world challenges such as sensor noise, asynchronous data communication, and limited computational resources.

6.2.1 Real word localization

This section evaluates the performance of the GBP. Figure 6.3 presents a portion of the trajectory followed by the swarm of robots, along with its corresponding estimated position. The proposed solution effectively localizes the robot with the error values summarized in Table 6.1. Despite the numerical errors, the approach demonstrates the ability to integrate information from odometry and UWB measurements while performing inference on a decentralized graph. This decentralized

framework successfully enables robot localization. Notably, the implementation of robust factors significantly enhances the localization algorithm's performance.

In Figure 6.4, which illustrates a portion of the dataset with the trajectories followed by each robot, it is evident that while the trajectories estimated using plain GBP appear smoother, the incorporation of robust factors yields better numerical accuracy. This improvement arises from the robust factors' ability to reject outlier contributions from certain graph branches, effectively disregarding the regularization term when necessary.

The benefits of robust factors are further highlighted in Figure 6.5, which depicts the cumulative distribution function (CDF) of localization errors. The robust factors demonstrate a clear advantage, significantly improving both performance and noise resilience. These findings underscore the potential of robust factors in enhancing the reliability and accuracy of decentralized localization solutions.

Algorithm / Robot	GBP		$GBP_{RobustFactor}$	
	RMSE	MAE	RMSE	MAE
TurteleBot1	36.8 Cm	30.8 Cm	25.5 Cm	19.3 Cm
TurteleBot2	40.5 Cm	36.9 Cm	30.8 Cm	27.0 Cm
TurteleBot3	29.5 Cm	27.5 Cm	20.1 Cm	16.5 Cm
TurteleBot4	32.6 Cm	27.7 Cm	22.9 Cm	16.8 Cm
Mean value	34.85 Cm	30.7 Cm	24.8 Cm	19.9 Cm

Table 6.1: Table of localization error

6.2.2 Isolated robot

In the previous experiment, all robots were able to perform UWB measurements with respect to the UWB antennas positioned on other robots and around the laboratory perimeter. This setup simulates a real-world scenario where a robot (TurtleBot1) operates in a GPS-denied environment or is unable to perform landmark-based measurements. Under these conditions, the robot relies solely on intra-robot measurements for relocalization while still performing UWB measurements relative to other robots.

The cumulative probability distributions of localization errors are presented in Figure 6.6, and the corresponding performance metrics are summarized in Table 6.2. While the localization error is relatively large, it is essential to consider that the

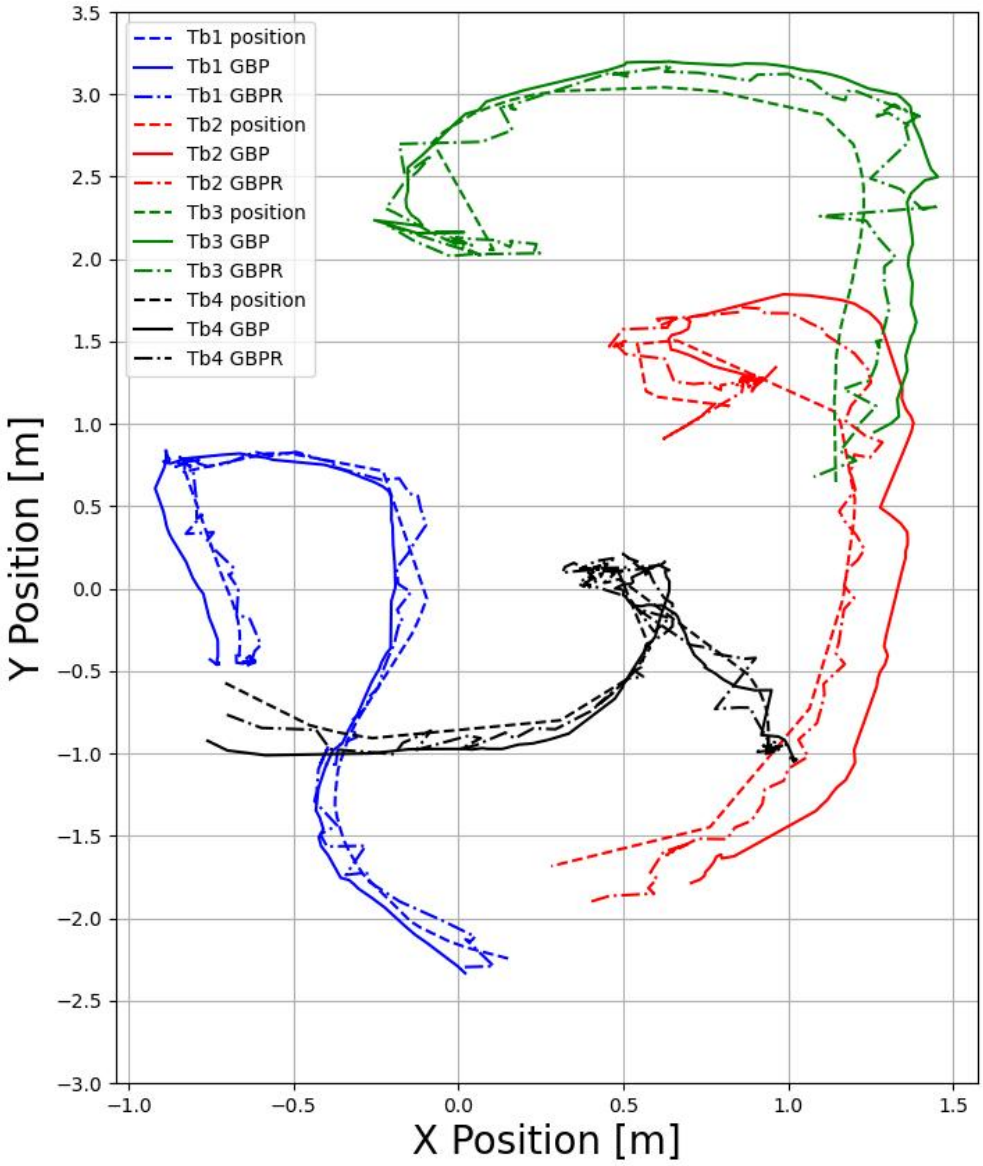
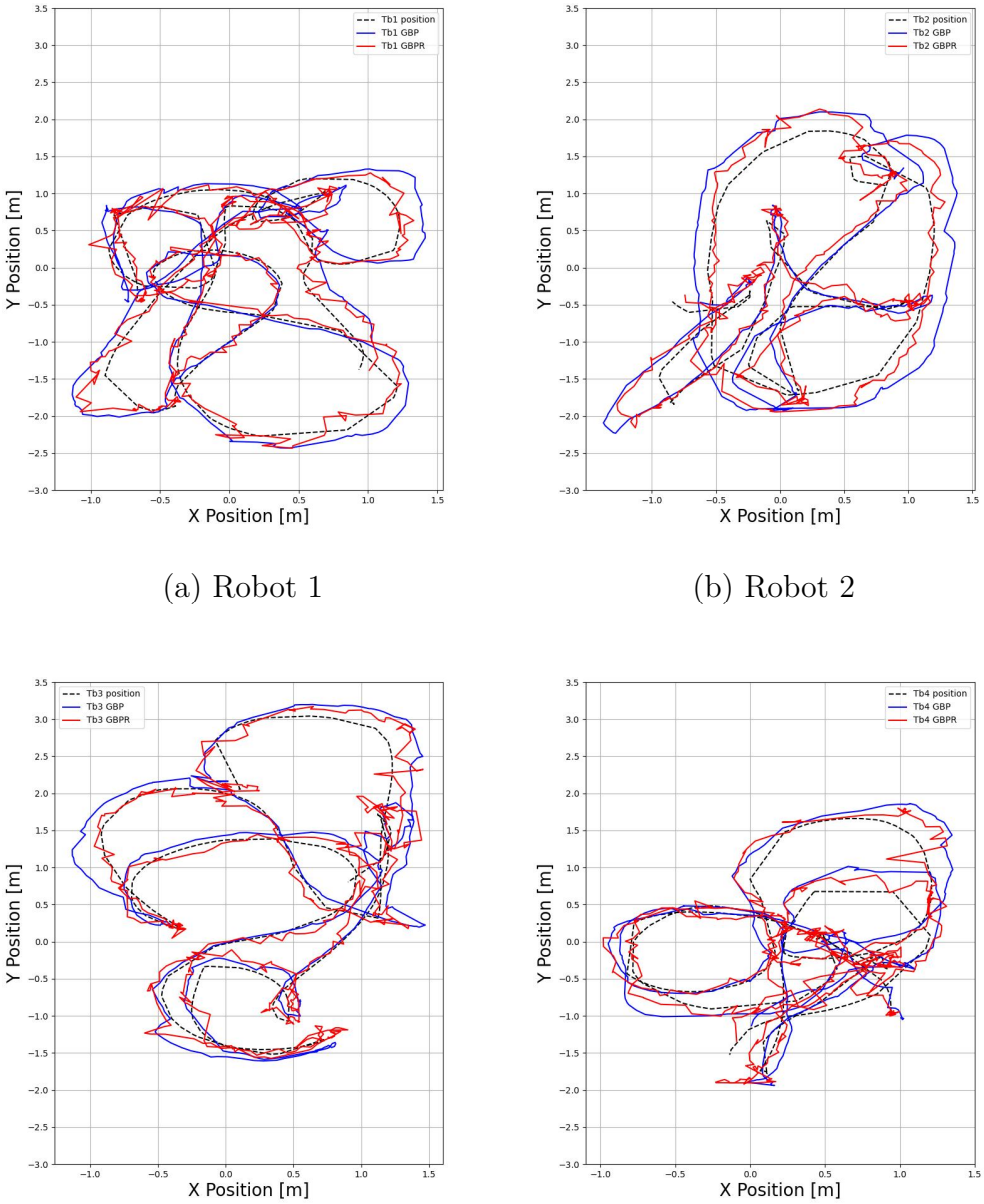


Figure 6.3: Swarm position and estimated position with and without robust factor



(a) Robot 1

(b) Robot 2

(c) Robot 3

(d) Robot 4

Figure 6.4: Overall caption for the 2×2 photo matrix.

robot in this scenario depends exclusively on three UWB measurements provided by moving anchors. The results once again underscore the effectiveness of robust

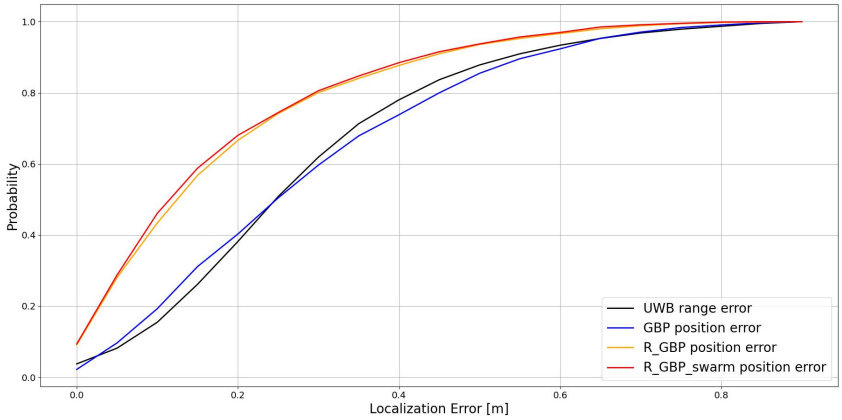


Figure 6.5: Cumulative distribution function comparing UWB range error, GBP and GBP with robust factor

factors in improving localization accuracy.As shown in Table 6.2, the introduction of robust factors slightly improves performance. Additionally, the impact of the isolated robot's localization errors on the other robots in the swarm, those capable of utilizing all sensors for localization, remains minimal. This experiment highlights the robot's ability to leverage data shared by the swarm to achieve localization, even under highly challenging conditions.

Algorithm / Robot	GBP		$GBP_{RobustFactor}$	
	RMSE	MAE	RMSE	MAE
TurteleBot1	60.8 Cm	57.9 Cm	56.8 Cm	51.4 Cm
TurteleBot2	44.3 Cm	38.3 Cm	34.6 Cm	28.6 Cm
TurteleBot3	30.5 Cm	27.0 Cm	21.05 Cm	17.5 Cm
TurteleBot4	36.3 Cm	31.4 Cm	25.7 Cm	19.8 Cm
Mean value	36.9 Cm	31.9 Cm	26.71 Cm	21.3 Cm

Table 6.2: Table of localization error with TurtleBot1 isolated

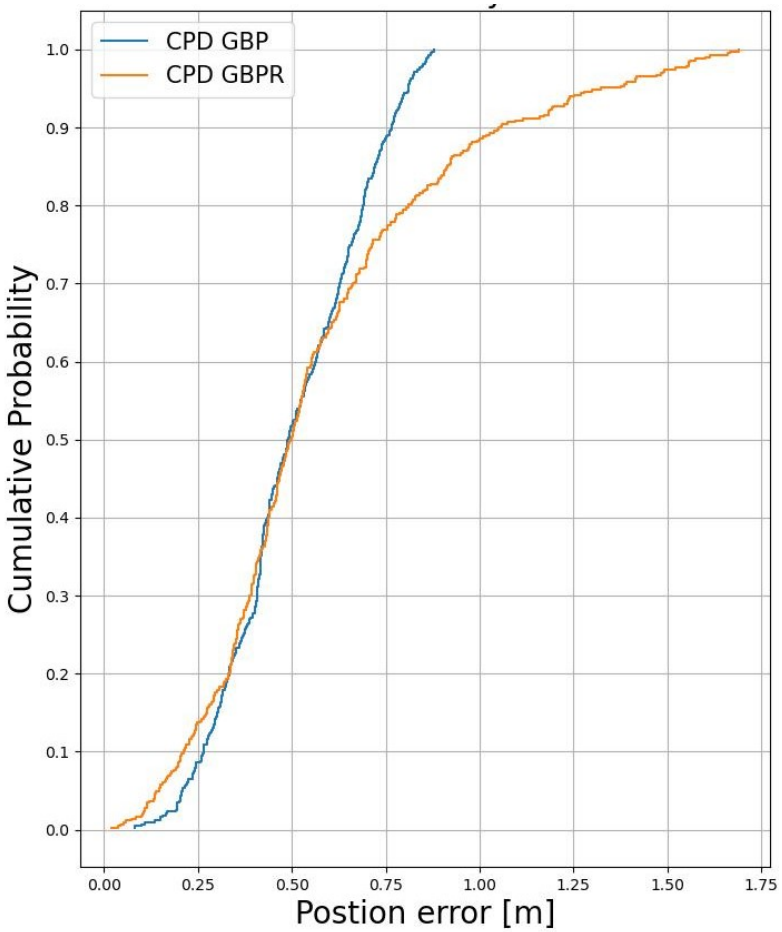


Figure 6.6: Cumulative distribution function comparing GBP and GBP with robust factor of an isolated robot

Chapter 7

Conclusion and Future Work

This chapter summarizes the key findings and contributions of the research. It discusses the main observations, emphasizing how the proposed solution tackles the challenges related to the GBP, factor graphs, and robot localization. Additionally, the chapter outlines the primary contributions of this work. Lastly, it explores potential directions for future research.

7.1 Summary of Findings

The primary objective of this thesis is to investigate the integration of factor graphs with the GBP algorithm and apply this approach to the problem of robot localization. The focus is on achieving localization in a fully decentralized manner, enabling robots to operate without synchronization.The main findings of this research can be summarized as follows:

1. **Development of a Factor Graph Model:** A factor graph was created to address both the specifics of the localization problem and the algorithm used for conducting inference. In particular, the design of the factor graph aimed to minimize the number of loops, which improves the convergence performance of the GBP algorithm.

2. **Implementation of the GBP Algorithm:** The GBP algorithm was implemented using the PyTorch library, leveraging specific insights from the factor graph model. This implementation achieves inference at a sufficiently high frequency, making it suitable for deployment on real-world robots operating on standard CPUs, rather than specialized processors designed for factor graph inference. The integration of new robust factor that integrate sensor specific knowledge that significantly improve the robustness to noise.

3. **Management of Non-Synchronized Robots:** The proposed solution effectively merges information from robots that share a common clock but do not localize synchronously, as often encountered in real-world robotics scenarios. This is achieved by interpolating data from other robots using a GPR and accurately modeling the factor graph to handle such asynchronous localization challenges.

7.2 Future Work

The integration of Gaussian Belief Propagation (GBP) within factor graph frameworks presents significant potential for advancing localization techniques in robotics, especially in decentralized systems. Possible future research to overcame the actual limitation of the factor graph model and GBP performance include:

1. **Multimodal Sensor Integration:** Investigate the fusion of heterogeneous sensor data into GBP-based factor graphs, enabling more accurate localization in complex environments. This includes developing strategies for compensating for individual sensor failures and improving the system's robustness.

2. **Robust Loss Functions:** Design and test innovative, robust loss functions that account for sensor-specific characteristics, such as time-of-flight for UWB and line-of-sight constraints for LiDAR. The goal is to reduce the impact of outliers, improve noise resilience, and ensure more reliable performance in real-world conditions.

3. **Dynamic and Adaptive Factor Graphs:** Develop factor graph frameworks that can adjust their factors in real-time according to environmental changes, sensor availability, and task-specific requirements. This adaptability will improve the system's performance in unpredictable environments.

4. **Leveraging Machine Learning for Data-Driven Graphs:** Use graph neural networks (GNNs) to guide the generation and adaptation of factor graphs based on observed data, allowing the system to identify key patterns and relationships automatically. This will enable robots to build and modify their localization models autonomously.

5. **Overcoming GBP Locality:** Explore methods to overcome the limitations of GBP locality by introducing global constraints or multi-level factor graphs. This will enable information propagation across distant nodes, enhancing the global coherence of the system's solution and improving scalability in larger, more complex environments.

6. **Hybrid Optimization Techniques:** I look forward to investigating the combination of GBP with global optimization methods to refine solutions after local convergence. This hybrid approach will allow for a more comprehensive optimization process, improving robotic systems' overall performance and adaptability.

This proposal aims to enhance the ability of robots to operate autonomously without relying solely on centralized systems or GPS; this work will pave the way for more autonomous and resilient robots with applications in areas such as exploration, disaster response, and agriculture.

7.3 Conclusion

The proposed solution, evaluated using real-world data and simulations demonstrated scalability and the capability to integrate odometry and UWB data for robot localization. It operates on a fully decentralized model, requiring minimal intra-robot data sharing, even in challenging scenarios where a robot relies solely on intra-robot measurements for localization.
The provided solution is capable, through using the PyTorch library or exploiting the computational capability of the CPU, of reaching operational frequency that makes it usable on real-world robots. The incorporation of a robust factor with an energy function tailored to the specific characteristics of the sensor significantly enhances localization performance.

The use of the GBP algorithm on fact graphs presents new opportunities for future research. Promising directions include exploring multimodality, developing novel, robust functions incorporating sensor knowledge to improve noise resilience, and enhancing factor graph frameworks with dynamic models and data-driven factors. Further advancements could investigate GBP's potential as a "glue" to enable simultaneous deployment of multiple optimization techniques on sub-portions of the same factor graph. These developments could facilitate autonomous model adaptability in dynamic environments.

Bibliography

[1] Frank Dellaert and Michael Kaess. 2017. DOI: 10.1561/2300000043 (cit. on pp. 3, 4, 12).

[2] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. «iSAM: Incremental Smoothing and Mapping». In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378. DOI: 10.1109/TRO.2008.2006706 (cit. on pp. 3, 5).

[3] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. «iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering». In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3281–3288. DOI: 10.1109/ICRA.2011.5979641 (cit. on p. 3).

[4] Yetong Zhang, Ming Hsiao, Jing Dong, Jakob Engel, and Frank Dellaert. «MR-iSAM2: Incremental Smoothing and Mapping with Multi-Root Bayes Tree for Multi-Robot SLAM». In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 8671–8678. DOI: 10.1109/IROS51168.2021.9636687 (cit. on p. 3).

[5] Lars A. A. Andersson and Jonas Nygards. «C-SAM: Multi-Robot SLAM using square root information smoothing». In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 2798–2805. DOI: 10.1109/ROBOT.2008.4543634 (cit. on p. 4).

[6] Vadim Indelman, Erik Nelson, Nathan Michael, and Frank Dellaert. «Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization». In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 593–600. DOI: 10.1109/ICRA.2014.6906915 (cit. on p. 4).

[7] Been Kim, Michael Kaess, Luke Fletcher, John Leonard, Abraham Bachrach, Nicholas Roy, and Seth Teller. «Multiple relative pose graphs for robust cooperative mapping». In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 3185–3192. DOI: 10.1109/ROBOT.2010.5509154 (cit. on pp. 5, 7).

[8] Trevor Halsted, Ola Shorinwa, Javier Yu, and Mac Schwager. *A Survey of Distributed Optimization Methods for Multi-Robot Systems*. 2021. arXiv: 2103.12840 [cs.RO]. URL: https://arxiv.org/abs/2103.12840 (cit. on p. 5).

[9] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues. «Multi-agent localization from noisy relative pose measurements». In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 364–369. DOI: 10.1109/ICRA.2011.5979799 (cit. on p. 5).

[10] Yulun Tian, Kasra Khosoussi, David M. Rosen, and Jonathan P. How. *Distributed Certifiably Correct Pose-Graph Optimization*. 2021. arXiv: 1911.03721 [math.OC]. URL: https://arxiv.org/abs/1911.03721 (cit. on p. 5).

[11] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P. How, and Luca Carlone. «Kimera-Multi: Robust, Distributed, Dense Metric-Semantic SLAM for Multi-Robot Systems». In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2022–2038. DOI: 10.1109/TRO.2021.3137751 (cit. on p. 5).

[12] Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Henrik I. Christensen, and Frank Dellaert. *Distributed Mapping with Privacy and Communication Constraints: Lightweight Algorithms and Object-based Models*. 2017. arXiv: 1702.03435 [cs.RO]. URL: https://arxiv.org/abs/1702.03435 (cit. on p. 6).

[13] Yulun Tian, Alec Koppel, Amrit Singh Bedi, and Jonathan P. How. «Asynchronous and Parallel Distributed Pose Graph Optimization». In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020), pp. 5819–5826. ISSN: 2377-3774. DOI: 10.1109/lra.2020.3010216. URL: http://dx.doi.org/10.1109/LRA.2020.3010216 (cit. on p. 6).

[14] Andrew J. Davison and Joseph Ortiz. *FutureMapping 2: Gaussian Belief Propagation for Spatial AI*. 2022. arXiv: 1910.14139 [cs.AI]. URL: https://arxiv.org/abs/1910.14139 (cit. on pp. 6, 7, 14, 16, 17).

[15] Joseph Ortiz, Talfan Evans, and Andrew J. Davison. *A visual introduction to Gaussian Belief Propagation*. 2021. arXiv: 2107.02308 [cs.AI]. URL: https://arxiv.org/abs/2107.02308 (cit. on p. 6).

[16] Riku Murai, Joseph Ortiz, Sajad Saeedi, Paul H. J. Kelly, and Andrew J. Davison. «A Robot Web for Distributed Many-Device Localization». In: *IEEE Transactions on Robotics* 40 (2024), pp. 121–138. ISSN: 1941-0468. DOI: 10.1109/tro.2023.3324127. URL: http://dx.doi.org/10.1109/TRO.2023.3324127 (cit. on p. 6).

[17] Aalok Patwardhan, Riku Murai, and Andrew J. Davison. «Distributing Collaborative Multi-Robot Planning With Gaussian Belief Propagation». In: *IEEE Robotics and Automation Letters* 8.2 (Feb. 2023), pp. 552–559. ISSN: 2377-3774. DOI: 10.1109/lra.2022.3227858. URL: http://dx.doi.org/10.1109/LRA.2022.3227858 (cit. on p. 6).

[18] Riku Murai, Ignacio Alzugaray, Paul H.J. Kelly, and Andrew J. Davison. «Distributed Simultaneous Localisation and Auto-Calibration Using Gaussian Belief Propagation». In: *IEEE Robotics and Automation Letters* 9.3 (Mar. 2024), pp. 2136–2143. ISSN: 2377-3774. DOI: 10.1109/lra.2024.3352361. URL: http://dx.doi.org/10.1109/LRA.2024.3352361 (cit. on pp. 6, 11).

[19] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. «Factor graphs and the sum-product algorithm». In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 498–519. DOI: 10.1109/18.910572 (cit. on p. 9).

[20] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, 2016. ISBN: 9781493938438. URL: https://books.google.it/books?id=kOXDtAEACAAJ (cit. on p. 13).