Imperial College London

Department of Computing

Distributed Coordination and Perception for Multi-Robot Systems

IMPERIAL

Aalok Patwardhan

September 2025

Supervised by Professor Andrew J. Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London.

ॐ

सा विद्या या विमुक्तये

sā vidyā yā vimuktaye

"Knowledge is that which leads to liberation."

— VISHNU PURANA 1.19.41

Statement of Originality

This thesis is my own work, except where otherwise indicated. All external sources and contributions have been appropriately referenced.

Abstract

This thesis explores a scalable, fully distributed framework for multi-robot collaboration under uncertainty in real-world environments. Multi-robot systems offer significant advantages over single-robot solutions, including parallelised task execution, redundancy, and scalability, enabling applications such as autonomous exploration and environmental monitoring. Achieving these benefits requires coordination, yet centralised approaches face bottlenecks, single points of failure, and limited adaptability. Real-world tasks are further complicated by noisy sensors, partial observations, and unreliable communication. Distributed frameworks overcome these challenges, enabling robots to reason locally and communicate with neighbours, supporting scalable and adaptive collective behaviour.

We formulate multi-robot coordination as probabilistic inference on factor graphs, using Gaussian Belief Propagation to perform local, asynchronous message-passing over nodes representing robot states and constraints. Our framework addresses complex multi-robot problems spanning multiple competencies such as path planning, exploration, and shape formation tasks. Robots coordinate to optimise locally, while emergent global behaviour arises from sparse but structured interactions. The framework also supports distributed consensus over continuous and discrete decision spaces, allowing swarms to agree on shared global states. Practical applicability is demonstrated via deployment on low-cost embedded hardware, achieving fully decentralised coordination through peer-to-peer communication. Finally, the thesis also demonstrates the applications of factor graphs to computer vision tasks, developing an uncertainty-aware rotation estimator that produces temporally consistent camera rotation estimates from RGB images.

This thesis demonstrates that factor-graph-based methods provide a lightweight and generalisable backbone for multi-robot systems, by unifying planning, exploration, formation, and consensus under a distributed, probabilistic approach. We envisage that these methods unlock the potential for large, heterogeneous swarms of the future to perform complex, coordinated behaviours in dynamic and uncertain environments.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Andrew J. Davison, for his guidance, support, and intuition. Andy, you have instilled in me the value of demo-oriented work and inspired my research on Gaussian Belief Propagation. I am fully convinced, thanks to your mentorship, that distributed robotics is the way of the future.

This journey would not have been possible without the support of my colleagues and friends, as well as Dyson Technology Ltd. I'd like to especially thank Iosifina Pournara for ensuring everything ran smoothly throughout. It has been a privilege to work alongside my peers in the lab; to be around such intelligence and banter has been a delight. Thank you to Riku Murai, Gwangbin Bae, Callum Rhodes, Marwan Taher, Kirill Mazur, Eric Dexheimer, Xin Kong, Ivan Kapelyukh, Ignacio Alzugaray-Lopez, Seth Nabarro, Hide Matsuki, Shikun Liu, and Joe Ortiz.

A special shout-out goes to my close friends outside university for keeping me grounded with random virtual coffee chats and reminders to enjoy life beyond work. I would also like to thank my past self from four years ago for having the courage to leave industry and take the plunge into a PhD. It was absolutely worth it.

Finally, this work would not have been possible without the love of my family: my Aai and Baba, my sister Rashmi, and all of my Ajis and Abas. Their support has been a constant source of strength.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It may be that machines will do the work that makes life possible and that human beings will do all the other things that make life pleasant and worthwhile.

— Isaac Asimov, Robot Visions

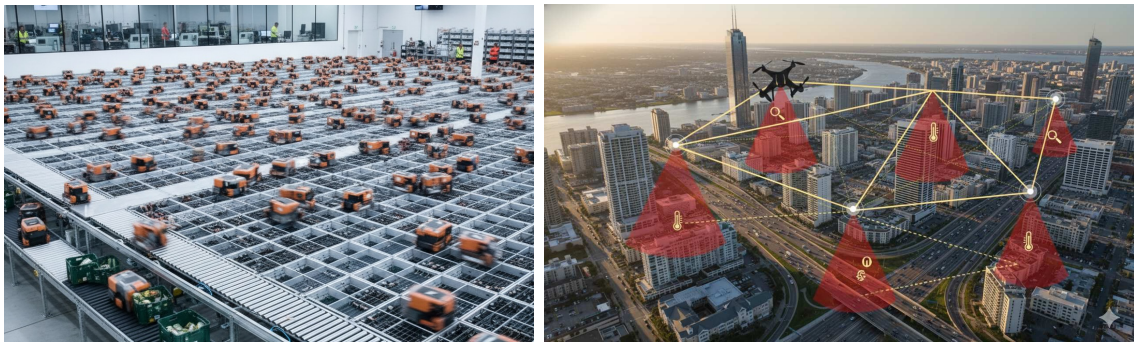
Contents

Innovation drives society forward, and yet it is increasingly our machines that turn ambitious ideas into reality. Tasks that once required teams of people can now be carried out by a handful of robots, and when robots are able to work together as a coordinated team they can achieve levels of efficiency and robustness that are difficult for humans alone.

This thesis is concerned with how such teams of robots can plan, share information, and make decisions in a distributed manner under uncertainty, and develops a probabilistic framework based on factor graphs and Gaussian Belief Propagation to support these capabilities across a range of multi-robot problems.

1.1 Multi-Robot Systems

Multi-robot systems are becoming increasingly prevalent in real-world applications, especially where it is too expensive or dangerous for humans to operate. The diverse range of fields span from warehouse logistics to environmental monitoring and disaster response.



(a) Modular grid robots in a grocery warehouse.

(b) Multi-robot environment monitoring in cities.

Figure 1.1: Multi-robot systems in real-world applications. *Images generated with AI.*

At Amazon fulfilment and sortation centres thousands of robots autonomously perform order picking and restocking; around 75% of Amazon parcels interact with these robotic

systems [businessinsider.com, 2025]. At Ocado up to 1000 grid-based robots are used to fetch storage bins containing fresh groceries for other robotic arms to handle the picking and packing [theengineer.co.uk, 2018] as shown in Figure 1.1a.

Underwater and aerial swarms enable data collection over large, inaccessible areas, which are quite often too dangerous for humans. Initiatives such as the ASTRIIS project deploy multiple gliders which coordinate to detect and track ocean fronts [ASTRIIS, 2023], and the EONIOS project uses a swarm of underwater vehicles to map and monitor artificial coral reefs [apnews.com, 2023].

Swarms of robots can be used to fight forest fires, or to perform autonomous search-and-rescue tasks in unknown terrains. Teams of drones in leader-follower formations have been able to efficiently survey large areas for mapping terrain and spreading of fire [Afghah et al., 2019], and recent swarm systems have shown success in real-time fighting and monitoring of wildfires in hazardous environments [globenewswire.com, 2024].

These multi-robot systems offer many advantages over single robot solutions to problems. Parallelisation of similar tasks can reduce mission time, and redundancy with homogeneous robots increases fault tolerance; robots are able to take over the responsibilities of their colleagues if and when necessary with minimal impact on the goal.

Centralised Systems. Often, a single computer or processor is responsible for coordinating the efforts of a team of robots. This involves path deconfliction, task scheduling, as well as information dissemination. Such centralised architectures are attractive because they offer a global view of the system state and can, in principle, compute globally consistent solutions for moderate team sizes. However, scaling such systems to large numbers of robots while maintaining centralised control introduces some key challenges:

- **Single point of failure.** If the central server or its network connection fails, global planning and coordination capabilities are lost, causing the robots to stop or crash, leading to costly downtime [Lienert et al., 2019].

- **Communication bottlenecks.** In centralised systems, all robots must communicate with a central controller, typically over wireless links with limited bandwidth. As the number of robots increases, the communication load scales accordingly, saturating the available channel capacity. Additionally, the central controller must process and coordinate a growing volume of information, resulting in increased computation and decision-making time [Gielis et al., 2022, Jamshidpey et al., 2025].

- **Lack of adaptability.** In dynamic or uncertain environments, centralised systems may struggle to respond quickly to local changes. Delays in propagating updated commands from the central controller to all robots can result in degraded performance or unsafe behaviour [Dias et al., 2006, Alonso-Mora et al., 2018b].

Decentralised Systems. A natural solution to these problems is to adopt a **decentralised** architecture, enabling local computation and autonomy. As each robot can reason and act based on its own state and that of its neighbours, such a system can expand to larger teams without a corresponding increase in global coordination overhead. A decentralised system is inherently more robust to communication loss: since coordination depends only on local interactions, partial failures in global communication degrade performance gradually rather than causing total system failure [Zhang et al., 2025, Halsted et al., 2021]. However, pure decentralisation introduces its own disadvantages:

- **Lack of global optimality:** Local optimisation can lead to suboptimal coverage, inefficient paths, or deadlocks, as robots pursue myopic (short-range) policies without full coordination [van den Berg and Overmars, 2005, van den Berg et al., 2009].

- **Convergence delays:** Consensus algorithms may require many iterations or fail to converge in large networks, poor connectivity, or adversarial conditions [Bizyaeva et al., 2023, Rogers and Gross, 2013].

Hybrid Systems. In practice, real-world systems often adopt **hybrid** designs that combine centralised high-level planning (mission objectives, global task allocation) with distributed local control (motion planning, collision avoidance) [St-Onge et al., 2020]. For example, a central controller may coordinate agents to efficiently collect and relay local observations, enabling global optimisation before disseminating updated instructions back to the fleet [Kapoutsis et al., 2021]. Recent work on multi-robot coordination presents architectures that can *dynamically* switch between more centralised joint planning within well-connected sub-teams and more decentralised behaviour for temporarily isolated robots, improving robustness to sensing and communication failures [Li et al., 2025].

Multi-robot problems vary in complexity, and the coordination scheme should ultimately be selected and defined according to the specific constraints and requirements of the problem at hand. Centralised approaches suit well-connected warehouses and moderate-scale settings, whereas distributed methods prioritise robustness and scalability over optimality for exploration. Hybrid methods provide a balanced trade-off for complex systems, particularly in heterogeneous fleets of robots where infrastructure is partial, communication is intermittent, or robot capabilities vary significantly.

And yet, even when the coordination schemes are precisely defined, the information available to robots is rarely so. Sensor data is noisy, incomplete, and varies across modalities. Effective decision-making requires reasoning under uncertainty, often combining multiple uncertain measurements and prior knowledge. To address this, we turn to a probabilistic representation of the multi-robot problem, explicitly modelling uncertainty and exploiting the underlying structure of the problem. The goal of this thesis is to develop a unified, distributed framework that is:

- **General**, supporting a range of tasks such as planning, mapping, and consensus.
- **Efficient**, minimising communication overhead and computation.

- **Practical**, able to run on resource-constrained embedded platforms in real time.
- **Probabilistically grounded**, enabling meaningful integration of noisy and incomplete data.

1.2 A Probabilistic Approach to Multi-Robot Problems

Real-world environments in which robots operate are inherently uncertain. For localisation, a robot might estimate its position by triangulating signals from satellites or radio beacons. It may also rely on local sensors such as LiDAR, RADAR, or infrared to build a map of its surroundings and localise within that map. Due to sensor noise and physical imperfections, these measurements are not perfectly repeatable and are better represented as samples from probability distributions (often Gaussian) with an associated mean and variance.

A single robot may only have partial knowledge of the global state. For example, a robot may make more accurate measurements in its immediate vicinity, with the measurement uncertainty increasing with distance from the robot. In a team performing environmental monitoring, each robot can only observe a limited region, contributing only a local perspective to the larger problem.

Communication can also be unreliable, whether in a team of robots sharing information with each other, or between a robot and its peripheral sensors or components. Information may be lost in transit due to real-world phenomena such as wireless interference, limited bandwidth, or multi-path effects. These issues further degrade the quality and reliability of information shared across the system.

To handle these challenges, each variable in an optimisation problem can be represented as a probabilistic distribution or a *belief*. Variables are often connected through con-

straints or measurements, but in most practical problems, these dependencies are *sparse*: each variable typically interacts with only a small subset of others. Probabilistic models allow robots to maintain and update uncertainty over time, fuse noisy measurements from heterogeneous sensors, and infer the values of unobserved variables based on known relationships.

1.2.1 Probabilistic Inference over Graphical Models

Probabilistic Graphical Models (PGMs) provide a flexible framework for reasoning under uncertainty. They allow us to model and calculate the joint probability distribution over complex systems of random variables by explicitly modelling their dependencies as a graph, making it easier to reason about structure. This makes them well-suited for inference in systems with noisy, incomplete, or distributed data.

Factor graphs are a type of probabilistic graphical model in which the joint probability distribution is broken down into a product of smaller functions, each depending on only a subset of variables. The graph is bipartite, consisting of variable nodes (representing unknown quantities to estimate) and factor nodes (representing constraints, measurements, or relationships between variables).

They are widely used in robotics, in Simultaneous Localisation And Mapping (SLAM), multi-robot coordination, and sensor fusion because they efficiently capture spatial and temporal dependencies. Their sparsity makes them well-suited to efficient inference, and their modular structure enables scalable and distributed computation.

There are two main approaches to inference in PGMs:

- **Maximum A Posteriori (MAP) inference** finds the mode of the variables given the observed data. This is a point estimate of the global state, but does not contain any information about the probabilistic uncertainty.

- **Marginal inference** on the other hand computes the probability distribution marginalised over each variable. This method retains information about the estimated values of the variables *along with their uncertainty* and as such is important for robotics applications where modelling and propagation of uncertainty is required.

In this thesis we focus on marginal inference through Belief Propagation (BP) [Pearl, 1982] which relies on local message-passing between variable and factor nodes in a factor graph. If the graph is tree-structured, BP converges to the true marginals in just two iterations of message passing. The method may not give exact marginals in graphs with loops however (loopy belief propagation) although it performs well in practice.

If all the variables and factors can be represented as Gaussian distributions, loopy belief propagation converges to the exact marginals. We refer to this as **Gaussian Belief Propagation (GBP)**, and it consists of three steps:

- Message passing from factors to variables.
- Belief update at each variable.
- Message passing from variables to factors.

GBP is ideal for real-time distributed inference in multi-robot systems, as the purely node-to-node message passing scales naturally to large networks. GBP has shown promising applications in recent work such as Robot Web [Murai et al., 2024b]. We explore GBP in more detail in Chapter 2.

1.3 Distributed Path Planning

While factor graphs and GBP provide a powerful abstraction, the real strength lies in their application to specific coordination problems. One such capability is distributed path planning — ensuring that multiple robots can move safely and efficiently in dynamic, shared environments.

In autonomous road junctions of the future, self-driving cars may traverse busy intersections at high speed. This would require precise coordination to avoid collisions. In search-and-rescue or exploration tasks robots may not only have to avoid collisions, but effectively plan paths in order to maximise information acquisition efficiently.

1.3.1 Single-Agent Path Planning

A robot navigating an environment with obstacles can perform path planning at a global or local level. Global planners require complete knowledge of the environment, often represented as an occupancy grid with information about obstacles, walls and free space. Using heuristic functions (such as minimising the total distance travelled), they can generate efficient paths from start to goal. This process is similar to solving a maze with a known layout, requiring systematic search strategies, whether through exhaustive grid-based methods [Hart et al., 1968], or by randomly sampling points in space [LaValle, 1998]). In dynamic environments, where the goal position may change or obstacles may be dynamic, these plans must be recomputed at every time step to remain valid and collision-free.

In contrast, local path planners operate in an 'online' manner, generating paths on-the-fly based on the robot's immediate surroundings. They require less computation and memory than global planners and can react quickly to nearby changes, making them particularly suitable for environments that change unpredictably. However, since the full

state of the environment is not known, local planners cannot guarantee that the final path from start to goal is globally optimal.

1.3.2 Multi-Agent Path Planning

When many robots share the same environment, the path planning problem for each robot becomes significantly more complex; the planned path of each robot effectively acts as a moving obstacle for all other robots. If robots each plan in an individualistic way without coordination, this can lead to conflicts such as deadlocks, where robots are unable to move as their planned paths block one another. This is especially true in constrained environments with obstacles. To avoid such situations, multi-robot planning algorithms must therefore optimise over the joint state-space of all robots and the environment.

Centralised solutions solve this by using a single processor with full knowledge of the states of all robots to compute collision-free trajectories. This allows for globally optimal solutions but quickly becomes intractable as the number of robots increases. To address this, techniques which constrain the problem can be used such as priority-based planning, where the path for one robot is computed first and treated as a dynamic obstacle for the next. However, these methods do not guarantee solutions, and require a central entity for conflict resolution.

In contrast, decentralised planners often operate locally, with each robot planning in its vicinity, and adjusting its path using information from neighbouring robots. These approaches may be purely reactive to observed changes, or can involve actively sharing planned paths to *negotiate* collision-free trajectories, and they are attractive as they require minimal computation and memory per-robot and are thus scalable. Communication can be direct (peer-to-peer), reactive, or stigmergic (leaving and sensing information in the environment). Still, coordination challenges remain such as avoiding deadlocks, ensuring globally optimal plans, and the need for decentralised conflict resolution. Despite

these limitations, decentralised planners are often well suited to highly dynamic environments where prioritising safety and adaptability outweighs the need for globally optimal solutions.

In Chapter 3 we introduce the GBP Planner, a distributed algorithm for multi-robot path planning. Each robot plans a path as a set of variables representing its position and velocity at discrete time steps in a short forward time window. The variables are dependent on each other via 'dynamics factors' representing constant-velocity model constraints. When two robots are near each other, they create collision avoidance factors between their planned paths. Robots only perform message passing with nearby robots, allowing them to optimise their paths based on purely local information.

Since we use GBP as the inference mechanism, the system can operate asynchronously and remain robust to communcation loss. It also scales well to larger teams of robots, as each robot only needs to communicate with its local neighbours.

While path planning is a core function, real-world tasks require robots to perform multiple interrelated behaviours, from exploring the environment to negotiating exploration goals. Achieving this integration calls for a higher-level coordination structure, which we address in the next section.

1.4 Towards Integrated Multi-Robot Behaviours

The various types of multi-robot interaction can be placed in a hierarchy of behaviours as shown in Figure 1.2, from individualistic actions at the base, to tightly integrated teamwork at the top level. Although applicable to any multi-robot problem, we use path planning.

Level 1 – Individualism. At the base level there is no collaboration: robots act individually to achieve their goals, unaware of the plans of other robots. These systems often

Figure 1.2: The hierarchy of multi-robot behaviours, illustrating increasing levels of information sharing and interdependence ranging from individualistic actions at the base to fully coordinated teamwork at the top.

use simple reactive collision-avoidance strategies, such as stopping or waiting if another robot is detected. Although straightforward to implement, this lack of coordination can lead to deadlocks, especially in cluttered environments.

Level 2 – Coordination. Robots still pursue individual goals, but can interact temporarily to prevent conflicts by adjusting their paths or speeds to allow each other to pass. This behaviour, similar to pedestrians side-stepping in a crowd, reduces the chance of deadlocks, while keeping decision-making local.

Level 3 – Collaboration. Robots have assigned individual goals or tasks informed by a higher-level group objective. For example, in an exploration scenario a robot's immediate goal might be to travel to an unexplored region, while the group's goal might be to minimse its total exploration time. This may require task swapping or path adjustments that slightly disadvantage an individual robot in the short term but improve the efficiency of the group as a whole. At this level, local and global objectives are linked, and efficiency is enabled through collaboration rather than individual optimisation.

Level 4 – Teamwork. At the highest level, robots act as a coordinated unit to achieve a shared goal, such as moving a heavy object together around obstacles. This requires tight coordination, collaboration, and continuous information exchange.

The higher we move in this hierarchy, the more important it becomes for each robot to

be aware of the plans and intentions of others. This seems at odds with the desire for decentralised systems, which avoid centralised control and rely only on local communication. Factor graphs and Gaussian Belief Propagation (GBP) offer a powerful way to bridge this gap: GBP enables each robot to iteratively refine its plan using only local information exchange. Over many iterations, these local updates can converge to solutions that achieve high levels of collaboration and teamwork without centralised control.

In complex multi-robot tasks, this often means optimising across many interdependent objectives or competencies, where progress in one affects the others. A unified, distributed approach using factor graphs allows robots to optimise independently across all competencies while still collaborating within specific ones.

In Chapter 4, we introduce the GBP Stack: a probabilistic framework for distributed optimisation in multi-robot, multi-objective tasks. We consider the problem where robots must collaboratively explore an environment, take noisy partial measurements of a global state, and simultaneously plan safe and smooth collision-free trajectories. These three interrelated sub-problems or competencies of path planning, exploration, and information acquisition are represented as a layers of variables and factors within a stacked factor graph (the GBP Stack).

Using GBP as the inference mechanism, each robot optimises the layers of its GBP Stack in coordination with the corresponding layers of neighbouring robots. When no neighbours are within communication range, or when coordination on certain layers is unnecessary or inefficient, a robot can continue optimising its own stack independently. This asynchronous, modular structure allows robots to adaptively engage in distributed optimisation only when and where it is beneficial. For example, a low-battery robot can prioritise planning and collision avoidance while skipping updates to less relevant higher layers.

1.5 Swarm Robotics and the Need for Consensus

Figure 1.3: In nature, swarms are made of many entities such as birds (left) or fish (right) that follow simple rules to create complex collective behaviour.

Large teams of robots operating together are often described as swarms, drawing inspiration from nature, where animals and insects exhibit complex collective behaviour through simple individual rules. In biological swarms, evolution has made individuals highly capable, yet they achieve remarkable collective behaviour using minimal communication. Ants coordinate using pheromones to forage and build with remarkable efficiency. Birds can react to their neighbours to create dynamic, shifting formations in flight, while schools of fish can move as one, rapidly reacting to predators or environmental changes.

These natural systems show that complex group behaviour can emerge from simple local interactions. In computer graphics, this principle was first demonstrated in the Boids algorithm [Reynolds, 1987], where just three basic rules of separation, alignment, and cohesion generated lifelike flocking behaviour. In robotics, similar principles have been demonstrated with small robots like Kilobots [Harvard University, 2014], where hundreds of tiny robots exhibit coordinated motion through local sensing and communication.

Contrary to the highly capable agents in biological swarms, much of the swarm robotics

literature treats robots as simple, often reactive entities, inspired by the idea that simple local rules are sufficient to generate emergent behaviour. However, this does not reflect the capabilities of modern robots, which are equipped with substantial onboard computation, sensing, and actuation. This raises the question: *can we combine the robustness and scalability of a swarm with higher-level reasoning at the individual level?*.

To enable this kind of collective intelligence, robots must be able to share information and make joint decisions. Often the task at hand involves reaching agreement on a shared global parameter [Jones and Hauert, 2024], such as the centre position of a shape formation, or choosing one option from a set of discrete alternatives [Valentini et al., 2017]; this process is known as consensus.

Consensus plays a central role in swarm robotics. In some tasks, robots must agree on a common state estimate derived from noisy and possibly conflicting local measurements. In others, they may need to synchronise their behaviour, choosing whether to explore, regroup, or switch formations. For example, in the application described in Chapter 4, robots explore an environment and must form a consensus on the estimated global state of the world, even when their individual observations differ.

Consensus problems can exist in both continuous and discrete decision spaces. In continuous spaces, robots might need to agree on a shared pose or direction of travel. In discrete spaces, they may need to agree on one of several predefined distinct behaviours. Literature has generally treated these two types of consensus problems separately.

In Chapter 5, we introduce DANCeRS, a unified consensus algorithm that handles both continuous and discrete decision spaces in a single framework. Each robot holds a GBP variable in a factor graph representing its belief about the global state. By constructing smoothness factors between their variables and those of their neighbours, robots are encouraged to converge on a shared belief.

We extend our GBP implementation to support optimisation over general Lie Groups,

allowing robots to reason over continuous state spaces such as the SE(2) group, which is critical for tasks involving positions and orientations. As a concrete example, we consider shape formation, where robots must arrange themselves into a known shape, but first need to agree on its global pose. We also introduce a novel distributed task-assignment method that enables robots to fill specific roles within that shape. In a second experiment, we show how the same algorithm can be used to solve the distributed best-of-N decision-making problem.

1.6 From Simulation to Hardware: Real-World Deployment

Having demonstrated our methods in simulation, we now focus on their practical implementation under real-world constraints. Gaussian Belief Propagation (GBP) relies on message passing between nodes in a factor graph and at a higher level, between robots. In simulation, we assume message passing to be instantaneous and lossless. Since the entire multi-robot system is simulated on a single processor, all agents share a common clock and operate in perfect synchronicity.

These assumptions do not hold in the real world. In a truly distributed system, each robot operates with its own clock, which can drift relative to others. Communication introduces latency and is prone to disruption; messages may be delayed, dropped, or received out of order due to wireless interference, multi-path effects, and bandwidth constraints.

In Chapter 6 we show for the first time a truly distributed implementation of GBP on edge devices demonstrating distributed consensus. While many so-called "distributed" robotics systems in literature still depend on central infrastructure or WiFi mesh networks with central routing, our implementation operates over peer-to-peer communication using the ESP-NOW protocol (based on the IEEE 802.11b standard).

Each robot comprises a low-cost Raspberry Pi processor and an ESP32 microcontroller. The ESP32 handles wireless communication via the ESP-NOW protocol, enabling low-latency peer-to-peer messaging between agents. The Raspberry Pi runs our lightweight GBP implementation and communicates with the ESP32 over a serial connection.

This setup offers a scalable and modular architecture for large robot swarms. Its low cost and compatibility with peripheral hardware such as motors, sensors, and cameras make it suited for real-world deployment in resource-constrained systems.

1.7 Factor Graphs for Perception: Rotation Estimation

So far, we have focused on coordination and control. Another important topic in robotics is visual perception, which can also be represented using factor graphs.

Accurately estimating camera rotation is critical for applications such as Augmented Reality (AR), visual SLAM systems, and scene understanding given visual cues. Images contain dense information about the scene, which can be utilised by robots whether through object-level segmentation, or lower level features widely used in SLAM.

Traditionally robots can use depth sensors and inertial measurement units (IMU) to help in state estimation, but these approaches require costly equipment and suffer from IMU drift. This is where visual state estimation shines. Provided that the image is well-textured, feature tracking can allow accurate rotation and translation estimation of a camera or robot, even in monocular settings [Matsuki et al., 2024], [Murai et al., 2024a]. Feature based methods are used to extract geometric priors such as vanishing points and lines. These methods struggle in texture-less environments.

In Chapter 7 we introduce a rotation estimator that uses densely-predicted surface nor-

mals to estimate the camera rotation with respect to the principal directions in the image. By considering each image frame's rotation estimates along with their covariance as variables in a factor graph, we enable temporally consistent estimation that functions well even in texture-less scenes, and does not require knowledge of camera intrinsics.

1.8 Contributions

This thesis establishes Gaussian Belief Propagation (GBP) as a unifying framework for distributed path planning, decision-making, and consensus in multi-robot systems, with both theoretical and practical impact. The principal contributions are:

1. **Gaussian Belief Propagation for distributed multi-robot planning and hierarchical decision-making.**

   This thesis presents the first application of GBP to distributed multi-robot path planning and introduces a hierarchical GBP framework for multifaceted multi-robot problems involving tightly coupled planning, estimation, and decision layers. Together, these approaches enable *scalable, fully decentralised coordination* through purely peer-to-peer message passing.

2. **Unified consensus and inference on nonlinear manifolds.**

   This thesis develops a single GBP-based inference framework that unifies consensus over (1) discrete decision spaces, as well as (2) continuous spaces. By extending GBP to state spaces over Lie Groups, the framework also supports complex decision and estimation problems on nonlinear manifolds, providing a principled approach for a wide range of multi-robot and geometric inference tasks.

3. **Practical deployment on low-cost robotic hardware.**

   The proposed methods are validated through the *first fully distributed demonstration of GBP-based planning and consensus on low-cost, resource-constrained*

robotic platforms, operating without central coordination, synchronised clocks or infrastructure-based communication.

4. **Robust rotation estimation from in-the-wild videos.**

   The thesis demonstrates a method for camera rotation estimation that does not require camera intrinsics, and works reliably on unconstrained, in-the-wild video data. This showcases the broad applicability and flexibility of probabilistic inference frameworks for challenging geometric estimation problems useful to robotics.

1.9 Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2.
We formally introduce the probabilistic inference methods and graphical models used in this thesis, and then proceed to derive Gaussian Belief Propagation, the core algorithm used in this thesis. Here, we also introduce Lie Groups and Lie Algebras, which are useful for representing non-Euclidean states such as rotations.

Chapter 3 – GBP Planner: Distributing Collaborative Multi-Robot Planning With Gaussian Belief Propagation.
In this chapter we present the work from our paper introducing a distributed algorithm for multi-robot coordination and path planning:

> Aalok Patwardhan, Riku Murai, Andrew J. Davison, **Distributing Collaborative Multi-Robot Planning With Gaussian Belief Propagation**, IEEE Robotics and Automation Letters (RA–L), 2023 [Patwardhan et al., 2023].

Robots plan for their states (position and velocity) at discrete time steps in a short forward time window. These states are the optimisation variables whose beliefs are shared with neighbouring robots via message-passing. Using GBP robots are able to jointly optimise their paths to avoid collisions and ensure smooth trajectories.

Chapter 4 – GBPStack: A Distributed Multi-Robot Framework for Exploration, Information Acquisition and Consensus.
This chapter presents the work from our paper:

> Aalok Patwardhan, Andrew J. Davison, **A Distributed Multi-Robot Framework for Exploration, Information Acquisition and Consensus**, IEEE International Conference on Robotics and Automation, 2024 [Patwardhan and Davison, 2024].

We argue that multi-robot teams must optimise over more complex problems spanning multiple competencies or sub-problems. Each of these competencies is represented in its own layer in a stack of factor graphs, enabling global optimisation to occur as and when necessary.

Chapter 5 – DANCeRS: A Distributed Algorithm for Negotiating Consensus for Robot Swarms.
This chapter presents the work in our paper presenting a distributed algorithm that uses GBP to optimise over consensus problems over discrete as well as continuous decision spaces:

> Aalok Patwardhan, Andrew J. Davison, **DANCeRS: A Distributed Algorithm for Negotiating Consensus in Robot Swarms**, arXiv, 2025 [Patwardhan and Davison, 2025].

Swarm formation-building is chosen as an example application where robots must form

a consensus on some global parameter, such as the position and orientation of a target formation. We also show how the same framework of negotiating consensus can be applied to discrete decision spaces, commonly referred to as the *best-of-N* problem.

Chapter 6 – A Fully Distributed Real-World Implementation of Gaussian Belief Propagation.
In this chapter we present our real-world implementation of a truly distributed multi-device system performing GBP for optimisation. We consider the same consensus problem from Chapter 5, and are motivated by real-time, low-cost and robust message passing and optimisation in a purely peer-to-peer manner without the need for any centrally-managed wireless or mesh network. We use a Raspberry Pi processor to perform the GBP algorithm and a ESP32 microcontroller to handle the physical transmission and reception of messages between devices. Our method is low cost, and can be applied to standard algorithms in robotics that use Python code.

Chapter 7 – U-ARE-ME: Uncertainty-Aware Rotation Estimation in Manhattan Environments. In this chapter we present the work from our paper:

> Aalok Patwardhan*, Callum Rhodes*, Gwangbin Bae, Andrew J. Davison, **U–ARE–ME: Uncertainty-Aware Rotation Estimation in Manhattan Environments**, International Conference on 3D Vision (3DV), 2025 (* Equal contribution) [Patwardhan et al., 2025].

Given an image we can predict the camera rotation relative to the global principal directions (the 'Manhattan Frame'), as well as the per-axis rotation uncertainty. Given a sequence of images or a video we construct a factor graph where the optimisation variables are the per-frame rotation estimation to enable a temporally consistent sequence of camera rotation. Our method is robust, and does not require camera intrinsics.

Chapter 8 – Conclusion and Future Directions. Here, we summarise the thesis contributions and discuss future research directions.

Chapter 2

Preliminaries

To make an apple pie from scratch,
you must first create the universe.

— Carl Sagan

Contents

This chapter covers the technical details used for the rest of the thesis. We primarily focus on methods for probabilistic inference, Probabilistic Graphical Models, Belief Propagation, and Lie theory.

2.1 Probabilistic Reasoning and Inference

In many real-world systems we must make decisions under uncertainty. Inference allows us to reason about unknown variables based on data that we can observe. This section introduces important concepts in probabilistic inference, beginning with Bayes' theorem, and leading to two key inference tasks: computing marginal distributions and estimating the most likely configuration of variables.

2.1.1 Bayes Theorem

Reverend Thomas Bayes (1701–1761) was an English minister and mathematician who was interested in the study of inverse probability: inferring the probability distribution of some unknown variable given the distribution of observed data. In his 1763 essay *An Essay towards solving a Problem in the Doctrine of Chances*, Bayes proposed a formal

method for updating beliefs about unobserved parameters in light of new evidence. Although it was published posthumously by his friend Richard Price, the essay laid the foundation for what is now known as Bayesian inference.

In contrast, frequentist statistics consider model parameters as fixed but unknown quantities and interpret probability as the limiting frequency of events over repeated trials. However, Bayesian methods treat parameters as random variables and allow for explicit incorporation of prior knowledge. This distinction is essential in robotics and machine learning, where reasoning under uncertainty and new information are fundamental.

The goal in many robotics problems is to estimate or reason about latent variables given some observed data. As an example, consider the problem of inferring a robot's true location in the world $\mathbf{x}$ given a sensor measurement $\mathbf{z}$. The core idea of using Bayesian inference is to assume we have some prior estimate of the probability distribution of the hidden parameter $\mathbf{x}$, and update it using the observations $\mathbf{z}$.

We are interested in computing the **posterior distribution** $p(\mathbf{x}|\mathbf{z})$: our belief about $\mathbf{x}$ after observing $\mathbf{z}$. Bayes theorem relates this posterior to two more quantities: the *prior* (of the hidden parameter) and the *likelihood* (of observing the data given our belief about the hidden parameter). Mathematically, Bayes Theorem is:

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})\ p(\mathbf{x})}{p(\mathbf{z})}\ , \tag{2.1}$$

where $p(\mathbf{x})$ represents the **prior**: our belief of the hidden parameter before any data is observed, $p(\mathbf{z}|\mathbf{x})$ is the **likelihood** of observing the data given a particular state $\mathbf{x}$, and $p(\mathbf{z})$ is known as the **evidence** which acts as a normalisation constant to ensure that the posterior integrates to one:

$$p(\mathbf{z}) = \int p(\mathbf{z}|\mathbf{x})\ p(\mathbf{x})\ . \tag{2.2}$$

In practice, Bayes Theorem allows us to fuse information from various sources and continuously update our beliefs about our model parameters in the light of new evidence.

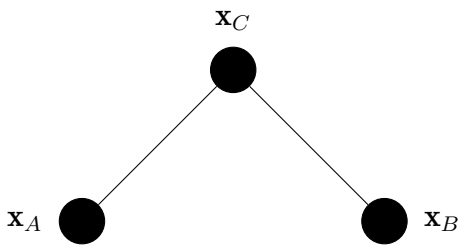


Figure 2.1: Variables $\mathbf{x}_A$ and $\mathbf{x}_B$ are conditionally independent to each other given a third variable $\mathbf{x}_C$. In this example diagram, edges between the variables denote a probabilistic relation.

2.1.2 Conditional Independence in Random Variables

Given a set of random variables $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, the joint probability distribution $p(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ defines the complete probabilistic nature of the system, as it represents the probability distribution over all possible assignments of the set $\mathbf{X}$. However, computing and representing such a joint distribution quickly becomes complicated as the number of variables in the set increases.

Two variables $\mathbf{x}_A$ and $\mathbf{x}_B$ are **independent** if their joint probability distribution is equal to the product of their marginal distributions:

$$\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \iff p(\mathbf{x}_A, \mathbf{x}_B) = p(\mathbf{x}_A) \cdot p(\mathbf{x}_B) \ . \tag{2.3}$$

However pure independence is rarely observed, and many times two variables are independent of each other, but each dependent on a third, common variable. This is known as **conditional independence**. For example as shown in Figure 2.1, if $\mathbf{x}_A$ and $\mathbf{x}_B$ are independent with respect to each other given a third variable $\mathbf{x}_C$, then the total joint probability distribution over the variables can be factored into smaller, more manageable factors:

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C) \cdot p(\mathbf{x}_C) \tag{2.4}$$

$$= p(\mathbf{x}_A|\mathbf{x}_C) \cdot p(\mathbf{x}_B|\mathbf{x}_C) \cdot p(\mathbf{x}_C) \ . \tag{2.5}$$

2.1.3 Maximum A Posteriori (MAP) Inference

Given some observations over a set of random variables, a common and useful computation step is to find the most probable configuration of all variables in the set, given some observations. This is called Maximum A Posteriori (MAP) inference, since we seek to compute the maximum of the posterior distribution of the variables $\mathbf{X}$ given some data $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_d\}$:

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} \; p(\mathbf{X}|\mathbf{Z}) \; . \tag{2.6}$$

Using Equation 2.1 to substitute for $p(\mathbf{X}|\mathbf{Z})$ the MAP estimate becomes

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} \; \frac{p(\mathbf{Z}|\mathbf{X}) \cdot p(\mathbf{X})}{p(\mathbf{Z})} \tag{2.7}$$

$$= \arg\max_{\mathbf{X}} \; p(\mathbf{Z}|\mathbf{X}) \cdot p(\mathbf{X}) \; , \tag{2.8}$$

sine $p(\mathbf{Z})$ is constant with respect to $\mathbf{X}$.

2.1.3.1 MAP Under Gaussian Distributions

In many robotics and perception problems, the likelihood $p(\mathbf{Z}|\mathbf{X})$ and prior $p(\mathbf{X})$ distributions can be assumed to be Gaussian in nature. As such, the posterior can be represented with an exponential of squared residuals $\mathbf{r}(\mathbf{X})$, weighted with a covariance matrix $\mathbf{\Sigma}$:

$$p(\mathbf{X}|\mathbf{Z}) \propto p(\mathbf{Z}|\mathbf{X}) \cdot p(\mathbf{X}) \tag{2.9}$$

$$\propto \exp\left(-\frac{1}{2}\boldsymbol{r}(\mathbf{X})^\top \mathbf{\Sigma}^{-1} \boldsymbol{r}(\mathbf{X})\right) \; , \tag{2.10}$$

where the vector of residuals $\boldsymbol{r}(\mathbf{X}) = \mathbf{z} - \boldsymbol{h}(\mathbf{X})$ is the difference between the measurements $\mathbf{z}$ and the values predicted by a linear measurement function $\boldsymbol{h}(\mathbf{X})$, and $\mathbf{\Sigma}$ is the covariance of the measurement model.

Taking the negative logarithm of the posterior, the MAP estimate from Equation 2.6 becomes

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \; -\log p(\mathbf{X}|\mathbf{Z}) = \arg\min_{\mathbf{X}} \; \frac{1}{2}\mathbf{r}^\top \mathbf{\Sigma}^{-1} \mathbf{r} = \arg\min_{\mathbf{X}} \; \frac{1}{2}\mathbf{r}^\top \mathbf{\Lambda} \mathbf{r} \; , \tag{2.11}$$

where $\mathbf{\Lambda} = \mathbf{\Sigma}^{-1}$ is the precision matrix of the measurement model, and we have used the shorthand $\mathbf{r} = \boldsymbol{r}(\mathbf{X})$ for the vector of residuals. The MAP inference problem then becomes equivalent to weighted non-linear least squares optimisation, which can be solved with classical optimisation techniques. It is important to note that by performing MAP inference we have obtained a *point estimate* of the set of random variables – the most probable configuration of all $\mathbf{x} \in \mathbf{X}$. However, we have lost all information regarding their covariances (uncertainties). We now introduce some key methods that can be used to solve non-linear least squares problems, and thus Gaussian MAP inference problems. In each case we seek to minimise an cost or 'energy' function $C(\mathbf{X})$:

$$C(\mathbf{X}) \ = \frac{1}{2}\mathbf{r}^\top \mathbf{\Lambda} \mathbf{r} \ . \tag{2.12}$$

As the cost function is non-linear, minimisation cannot occur in closed form and as such we turn to iterative methods that incrementally update an estimate $\mathbf{X}_t$ with the goal of the cost function converging to a minimum.

2.1.3.2 Gradient Descent

This method uses the first-order Taylor series expansion of the cost function $C(\mathbf{X})$ to iteratively update the variables $\mathbf{X}$ in the direction of steepest descent.

$$\mathbf{X}_{t+1} = \mathbf{X}_t \ + \boldsymbol{\delta} \ , \tag{2.13}$$

$$\boldsymbol{\delta} = - \ \alpha \nabla C(\mathbf{X}_t) \ , \tag{2.14}$$

with the gradient of the cost function computed as:

$$\nabla C(\mathbf{X}) \ = \ \mathbf{J}^\top \mathbf{\Lambda} \mathbf{r} \ , \tag{2.15}$$

where the shorthand $\mathbf{J} = \boldsymbol{J}(\mathbf{X})$ denotes the Jacobian matrix of the residual function, and α is a hyper-parameter known as the learning rate. This process is repeated until a stopping criteria is reached, for example if the update $\boldsymbol{\delta}$ at a time step is smaller than some predefined threshold.

Gradient descent has found popularity in the world of Deep Learning, due to the efficiency of computing gradients in neural networks using back-propagation, and scalability to extremely high-dimensional problems. However in nonlinear least squares problems it often converges slowly especially if the cost landscape is highly curved or ill-conditioned.

2.1.3.3 Gauss-Newton (GN) Algorithm

Second-order methods greatly improve convergence by making use of the Hessian of the cost function $\boldsymbol{H}_C(\mathbf{X})$:

$$\boldsymbol{H}_C(\mathbf{X}) = \nabla^2 C(\mathbf{X}) = \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} + \sum_i (\boldsymbol{\Lambda}\mathbf{r})_i \cdot \nabla^2 \mathbf{r}_i \tag{2.16}$$

$$\approx \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} \ . \tag{2.17}$$

The second-order derivative term in Equation 2.16 uses the second derivatives of the residuals $\boldsymbol{r}_i(\mathbf{X})$ and may be costly to compute; the Gauss-Newton method approximates the Hessian by ignoring these terms, assuming that the residuals are small near the optimum.

The approximate second-order Taylor series expansion of the cost function now becomes:

$$C(\mathbf{X}_t + \boldsymbol{\delta}) \approx C(\mathbf{X}_t) + \nabla C(\mathbf{X})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \nabla^2 C(\mathbf{X})\boldsymbol{\delta} \ . \tag{2.18}$$

Substituting Equations 2.15 and 2.17 into this equation, and setting its derivative with respect to $\boldsymbol{\delta}$ to zero yields:

$$\frac{\mathrm{d}C(\mathbf{X}_t + \boldsymbol{\delta})}{\mathrm{d}\boldsymbol{\delta}} \approx \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{r} + \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} \ \boldsymbol{\delta} = 0 \ , \tag{2.19}$$

(where we have omitted the argument $\mathbf{X}$ of the functions for brevity) leading to the linear system of equations known as the *normal equations*:

$$\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} \ \boldsymbol{\delta} = -\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{r} \ . \tag{2.20}$$

This system of equations can be solved to obtain the update amount $\boldsymbol{\delta} = -(\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J})^{-1} \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{r}$ to be used in the next iteration. The Gauss-Newton algorithm works by approximating the curvature of the cost function $C(\mathbf{X})$ by using a local approximation to the true Hessian, and produces faster convergence in optimisation than Gradient Descent. However this method may diverge if the initial solution is far from the minimum, or if $\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J}$ is ill-conditioned.

2.1.3.4 Levenberg-Marquardt (LM) Algorithm

This method combines the fast convergence of Gauss-Newton with the stability of Gradient Descent. A damping term is added to the approximation of the Hessian to modify Equation 2.20:

$$(\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} + \lambda \mathbf{I}) \boldsymbol{\delta} = -\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{r} \ , \tag{2.21}$$

where $\lambda \geq 0$ is a damping parameter, and $\mathbf{I}$ is the Identity matrix. At each iteration λ is adjusted based on the change in the cost function; if the cost increases, λ is increased and if the cost decreases, λ is reduced.

The Levenberg–Marquardt algorithm smoothly interpolates between gradient descent and Gauss–Newton. When λ is large, the update direction approximates $-\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{r}$, resembling Gradient Descent with a small learning rate. The algorithm behaves like Gauss-Newton when λ approaches 0. This adaptive behaviour makes LM particularly effective for highly nonlinear problems or when the initial estimate is far from the optimal solution. This method is used to solve the optimisation problem in Chapter 7.

2.1.4 Marginal Inference

Whereas Maximum A Posteriori (MAP) inference provides a point estimate of the most probable configuration (values) of the variables in a joint probability distribution, it does

not provide information about the confidence of that point estimate. However knowing this confidence or uncertainty is useful for making reliable decisions in robotics.

The goal in marginal inference is to compute the probability distribution over a single variable (or a subset of variables) $\mathbf{X}_A \subset \mathbf{X}$. This is done by integrating the joint probability distribution over all *other* variables:

$$p(\mathbf{X}_A) = \int_{\mathbf{X}_{\setminus A}} p(\mathbf{X}) \, \mathrm{d}\mathbf{X}_{\setminus A} \ , \tag{2.22}$$

where $\mathbf{X}_{\setminus A}$ denotes the variables not in the set $\mathbf{X}_A$.

Although Equation 2.22 may be intractable in general, when the joint probability distribution $p(\mathbf{X})$ is (multi-variate) Gaussian, the marginal distributions have closed-form solutions and are also Gaussian.

Consider a joint probability distribution over a set of variables $\mathbf{X}$:

$$p(\mathbf{X}) = \mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \ , \tag{2.23}$$

where $\mathbf{X} \in \mathbb{R}^n$, $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ are the mean vector and covariance matrix respectively.

If we are interested in finding the marginal distribution over the set X_A, we can first partition $\mathbf{X}, \boldsymbol{\mu}$, and Σ as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_A \\ \mathbf{X}_{\setminus A} \end{bmatrix} \ , \ \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_{\setminus A} \end{bmatrix} \ , \ \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{A\ A} & \boldsymbol{\Sigma}_{A\ \setminus A} \\ \boldsymbol{\Sigma}_{\setminus A\ A} & \boldsymbol{\Sigma}_{\setminus A\ \setminus A} \end{bmatrix} \ . \tag{2.24}$$

The marginal distribution over X_A is obtained by simply extracting the relevant slice of the joint mean and covariance matrix:

$$p(\mathbf{X}_A) = \mathcal{N}(\mathbf{X}_A; \ \boldsymbol{\mu}_A, \boldsymbol{\Sigma}_{AA}) \ . \tag{2.25}$$

2.1.4.1 Relation to MAP Inference

In many estimation problems we begin by finding the MAP solution $\mathbf{X}^*$ by minimising a cost function $C(\mathbf{X})$ [Bishop, 2006]. However since MAP only gives a point estimate, we wish to characterise the uncertainty around $\mathbf{X}^*$. We assume the underlying posterior is locally Gaussian around the MAP solution, this is known as Laplace's approximation:

$$p(\mathbf{X}|\mathbf{Z}) \propto \exp(-C(\mathbf{X})) \propto \mathcal{N}(\mathbf{X}^*, \boldsymbol{\Sigma}) \ . \tag{2.26}$$

Performing a second order Taylor series expansion of $C(\mathbf{X})$ around $\mathbf{X}^*$ we arrive at:

$$C(\mathbf{X}) \approx C(\mathbf{X}^*) + \nabla C(\mathbf{X}^*)(\mathbf{X} - \mathbf{X}^*) + \frac{1}{2}(\mathbf{X} - \mathbf{X}^*)^\top \mathbf{H}(\mathbf{X} - \mathbf{X}^*) \ , \tag{2.27}$$

where $\mathbf{H} = \nabla^2 C(\mathbf{X}^*)$ is the Hessian of the cost function evaluated at the MAP solution $\mathbf{X}^*$. Under Laplace's approximation, the gradient of the cost function around the MAP solution is zero, so substituting for $C(\mathbf{X})$ in Equation 2.26 with $\nabla C(\mathbf{X}^*) \approx 0$ we get:

$$p(\mathbf{X}|\mathbf{Z}) \propto \exp\left(-\frac{1}{2}(\mathbf{X} - \mathbf{X}^*)^\top \mathbf{H}(\mathbf{X} - \mathbf{X}^*)\right) = \mathcal{N}(\mathbf{X};\ \mathbf{X}^*, \mathbf{H}^{-1}) \ . \tag{2.28}$$

Therefore, given an MAP solution $\mathbf{X}^*$ to a non-linear least-squares problem, we can approximate the posterior covariance $\boldsymbol{\Sigma}$ by inverting the Gauss-Newton approximation of the Hessian of the cost function at $\mathbf{X}^*$:

$$\boldsymbol{\Sigma} \approx \mathbf{H}^{-1} \approx (\mathbf{J}^\top \mathbf{J})^{-1} \ , \tag{2.29}$$

where $\mathbf{J}$ is the Jacobian of the residual function $\boldsymbol{r}(\mathbf{X})$ evaluated at $\mathbf{X}^*$.

Marginal inference over a desired subset $\mathbf{X}_A$ of variables is then simplified to extracting the $\boldsymbol{\Sigma}_{AA}$ submatrix. Approximating the covariance $\boldsymbol{\Sigma}$ incurs minimal extra cost, as the Jacobian $\mathbf{J}$ is calculated during the iterative MAP optimisation methods such as Gauss-Newton or Levenberg-Marquardt.

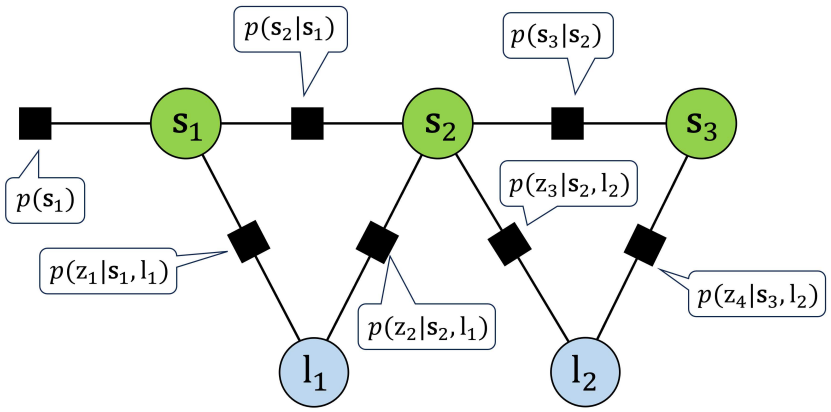


Figure 2.2: A Probabilistic Graphical Model (PGM) of a simple SLAM problem with landmarks $\mathbf{l}_1, \mathbf{l}_2$ observed from robot poses $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$, showing the sparse structure of the probabilistic relationships (squares) between variables (circles).

2.2 Probabilistic Graphical Models and Factor Graphs

Many probabilistic models in MAP estimation and marginal inference exhibit structure in the form of conditional independencies between subsets of variables. Probabilistic Graphical Models (PGMs) make this structure explicit by representing variables as nodes and probabilistic relationships as edges. For example, consider a simple SLAM scenario with three robot poses $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}$ and two landmarks $\mathbf{L} = \{\mathbf{l}_1, \mathbf{l}_2\}$, shown in Figure 2.2. The observations of landmark $\mathbf{l}_1$ from $\mathbf{s}_1$ and $\mathbf{s}_2$ as well as observations of landmark $\mathbf{l}_2$ from $\mathbf{s}_2$ and $\mathbf{s}_3$ are denoted as $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4\}$ respectively. There is also a prior on the first pose $\mathbf{s}_1$, and motion likelihoods between consecutive poses. The joint probability distribution over the poses, landmarks and observations factorises as:

$$p(\mathbf{S}, \mathbf{L}, \mathbf{Z}) = p(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{l}_1, \mathbf{l}_2, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4) \tag{2.30}$$

$$= p(\mathbf{s}_1)\ p(\mathbf{s}_2 \mid \mathbf{s}_1)\ p(\mathbf{s}_3 \mid \mathbf{s}_2) \tag{2.31}$$

$$\times\ p(\mathbf{z}_1 \mid \mathbf{s}_1, \mathbf{l}_1)\, p(\mathbf{z}_2 \mid \mathbf{s}_2, \mathbf{l}_1) \tag{2.32}$$

$$\times\ p(\mathbf{z}_3 \mid \mathbf{s}_2, \mathbf{l}_2)\, p(\mathbf{z}_4 \mid \mathbf{s}_3, \mathbf{l}_2). \tag{2.33}$$

This factorisation naturally leads to a **factor graph** representation, where each factor corresponds to either a motion term, a pose or landmark prior, or an observation.

2.2.1 Factor Graphs

Formally, a factor graph represents the factorisation of a general joint probability distribution $p(\mathbf{X})$ over a set of variables $\mathbf{X}$ into positive potential functions f_s which depend on subsets (cliques) $\mathbf{X}_s$ of all variables. It is an undirected bipartite graph of variable nodes and factor nodes that represent the potential functions f_s. An edge exists between a variable node $\mathbf{x}_i$ and a factor node f_j if $\mathbf{x}_i$ is an argument of the function f_j. As such the joint distribution can be represented as:

$$p(\mathbf{X}) \propto \prod_s f_s(\mathbf{X}_s) \ , \tag{2.34}$$

where each f_s is a local function of a clique of variables $\mathbf{X}_s$.

Factor graphs are particularly well-suited to inference algorithms like Belief Propagation, where messages pass between variables and factors, and the overall connectivity of the nodes in the graph is sparse. In Gaussian factor graphs, this sparsity enables efficient marginal inference, forming the foundation for Gaussian Belief Propagation (GBP).

2.2.1.1 Gaussian Factor Graphs

In a Gaussian factor graph, each factor f_s describes a constraint or measurement taking the form of a Gaussian distribution [Davison and Ortiz, 2019]:

$$f_s(\mathbf{X}_s) = K e^{-\frac{1}{2}\left[(\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s))^\top \boldsymbol{\Sigma}_s^{-1} (\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s))\right]} \ , \tag{2.35}$$

where $\boldsymbol{h}_s(\mathbf{X}_s)$ is the (possibly non-linear) functional form of the measurement or constraint over the set of variables $\mathbf{X}_s$ that the factor represents, $\boldsymbol{z}_s$ is its observed or expected value, $\boldsymbol{\Sigma}_s$ is the covariance of the constraint and K is a constant.

Taking the negative logarithm of the joint distribution (which is a product of many factors) yields a cost function:

$$C(\mathbf{X}) = -\log p(\mathbf{X}) = \frac{1}{2}\sum_{s}(\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s))^\top \boldsymbol{\Lambda}_s(\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s)) + L \ , \tag{2.36}$$

where L is a constant that can be ignored during optimisation.

Factor graph inference, for instance in probabilistic estimation problems such as SLAM, involves finding the values of variables $\mathbf{X}$ which maximise the product $p(\mathbf{X})$. Equation 2.36 shows that inference over a Gaussian factor graph amounts to finding $\mathbf{X}$ to *minimise* $C(\mathbf{X}) = -\log p(\mathbf{X})$, or in other words, minimising a sum of squared residuals.

This is an incredibly powerful connection:

> **Any problem that can be represented as a nonlinear least-squares optimisation problem can also be viewed as performing probabilistic inference on a Gaussian factor graph.**

This opens up the use of the many factor graph inference tools originally developed for probabilistic inference. This was highlighted in [Mukadam et al., 2018b, Dellaert, 2021], but previously only demonstrated for single robot planning using centralised solvers such as GTSAM [Dellaert, 2021]. In this thesis, we use factor graphs to model a wide range of multi-robot coordination problems as well as temporally consistent rotation estimation for computer vision tasks.

2.3 Gaussian Belief Propagation

The rise of low-power and low-cost computing platforms has transformed the landscape of modern computing, especially through how intelligent systems are designed and deployed. The Internet of Things (IoT) has meant that modern environments are filled with smart

devices which have onboard compute, sensing and communication capabilities. These devices are typically heterogeneous, and communicate with each other in a peer-to-peer manner, operating asynchronously without the need for a centralised controller.

In these settings global coordination is infeasible; devices must rely on local computation, message passing with neighbours, and must maintain robustness to partial or unreliable information. To support scalable and distributed inference under these constraints, we turn to iterative algorithms like Belief Propagation, which naturally align with the structure of factor graphs.

Belief Propagation (BP) was introduced in [Pearl, 1982], and is a method for performing distributed inference over factor graphs. It calculates the marginals of the joint distribution, and operates by passing messages between nodes. BP leverages local computation to perform inference globally.

A **message** in belief propagation simply represents a probability distribution summarising a node's belief about a variable. For example, a Gaussian probability distribution in n dimensions could be represented as a message $\mathbf{m} = \{\boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}\}$, where μ is the mean vector of the distribution, and $\boldsymbol{\Sigma}$ is the covariance matrix. This means that the messages passed between nodes are small in size, consisting of a low-dimensional vector and matrix. In practice and for reasons that will be discussed, in this thesis we instead use the *Information* form of representing Gaussian distributions with the components of a message being $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$, and the message size remains unchanged.

Following [Davison and Ortiz, 2019], an iteration of BP occurs in three steps:

1. A **variable to factor** message encodes the variable's belief: the marginal probability distribution of its own state.
2. A **factor to variable** message represents that factor's interpretation about the receiving variable's marginal distribution (belief), having integrated the information it has collected.

3. In the **belief update** step, a variable combines messages received from its connected factors to update its belief.

In a tree or chain structure, BP converges to the true marginals in one iteration of message passing (from the root node to the leaves of the tree or end of the chain, and back). An advantage of BP is that the order of message passing does not matter, and it can occur synchronously (all nodes send outgoing messages at the same time), or asynchronously in various message passing schedules. The fact that each node can perform computation independently highlights the scalability and parallel compute capabilities of the algorithm.

If there are one or more cycles in the factor graph, BP takes the form of Loopy Belief Propagation (LBP) which has empirically produced good results [Murphy et al., 1999]. However an issue with LBP is that it lacks convergence guarantees, and if it does converge, it is not guaranteed to converge on to the true marginals.

In Gaussian Belief Propagation (GBP), all probability distributions are Gaussian, and all factors are Gaussian functions of their connected variables. Unlike LBP, if GBP converges, it converges to the true marginals, and has shown stronger empirical performance. The actions of marginalisation and conditioning over Gaussian distributions are closed form, leading to simpler implementation. We now derive the steps of GBP in detail, beginning with our choice of representation of the Gaussian distributions.

A Gaussian distribution $p(\mathbf{X}) \propto \exp^{-C(\mathbf{X})}$ can be written in exponential form where the exponent uses a quadratic cost function $C(\mathbf{X})$ which can be written in one of two equivalent forms:

$$C(\mathbf{X}) = \underbrace{\frac{1}{2}(\mathbf{X}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{X}-\boldsymbol{\mu})}_{\textbf{Moments form}} \tag{2.37}$$

or

$$C(\mathbf{X}) = \underbrace{\frac{1}{2}\mathbf{X}^\top \boldsymbol{\Lambda}\mathbf{X} - \boldsymbol{\eta}^\top \mathbf{X}}_{\textbf{Information form}} \tag{2.38}$$

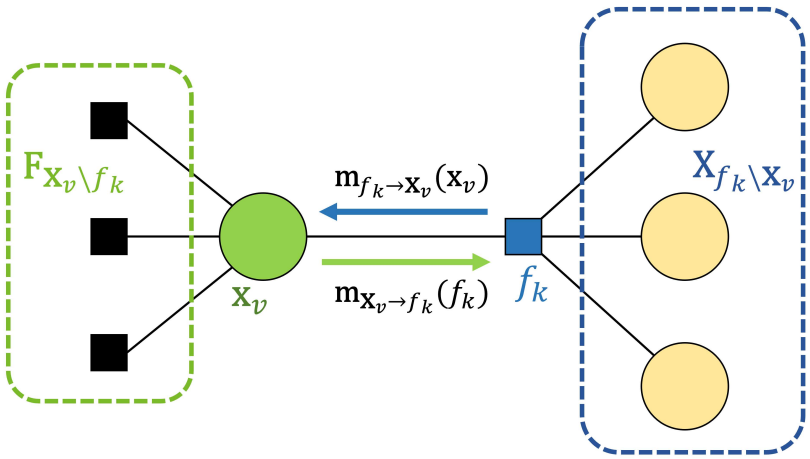


Figure 2.3: An example factor graph with variables shown as circles, and factors shown as squares. Here the focus is on the connection between variable $\mathbf{x}_v$ and factor f_k; the variable to factor message and factor to variable messages are denoted as $\mathbf{m}_{\mathbf{x}_v \to f_k}(f_k)$ and $\mathbf{m}_{f_k \to \mathbf{x}_v}(\mathbf{x}_v)$ respectively. $\mathbf{F}_{\mathbf{x}_v \setminus f_k}$ is the set of neighbouring factors of variable $\mathbf{x}_v$ excluding f_k, and $\mathbf{X}_{f_k \setminus \mathbf{x}_v}$ is the set of neighbouring variables of factor f_k excluding $\mathbf{x}_v$.

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and covariance of the distribution respectively, $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the inverse covariance known as the precision matrix, and $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$ is the information vector. Intuitively where $\boldsymbol{\mu}$ denotes the mean value of $\mathbf{X}$, $\boldsymbol{\eta}$ can be thought of as the mean weighted by its uncertainty. The Gaussian distribution can be written in both Moments and Information form as:

$$\underbrace{\mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}_{\textbf{Moments form}} = \underbrace{\mathcal{N}^{-1}(\mathbf{X}; \boldsymbol{\eta}, \boldsymbol{\Lambda})}_{\textbf{Information form}} , \tag{2.39}$$

In this thesis we use the Information form, as it simplifies the computation involved in conditioning and forming products over probability distributions, needed for forming the posterior.

In GBP, we consider a factor graph $\mathcal{G}(\mathbf{X}, \mathbf{F})$ with N_V variables and N_F factors. Variables $\mathbf{X} = \{\mathbf{x}_v\}_{v \in \{1,2,\ldots,N_V\}}$ are assumed to be Gaussian; each variable $\mathbf{x}_v$ has a belief $b(\mathbf{x}_v) = \mathcal{N}^{-1}(\mathbf{x}_v;\ \boldsymbol{\eta}_v, \boldsymbol{\Lambda}_v)$. Factors $\mathbf{F} = \{f_k\}_{k \in \{1,2,\ldots N_F\}}$ are Gaussian constraints between variables; a factor $f_k(\mathbf{X}_{f_k})$ is an arbitrary function of a set of variables $\mathbf{X}_{f_k}$, and it may be non-linear. For the rest of this chapter, $\mathbf{X}_{f_k} \subseteq \mathbf{X}$ denotes the set of variables directly connected to (neighbours of) factor f_k, and $\mathbf{F}_{\mathbf{x}_v} \subseteq \mathbf{F}$ denotes the set of factors that are neighbours of

variable $\mathbf{x}_v$. An example minimal factor graph is shown in Figure 2.3 as a reference for the following sections.

2.3.1 Variable Belief Update

A variable $\mathbf{x}_v$ updates its belief by taking the product of all incoming messages from its connected factors:

$$b(\mathbf{x}_v) = \mathcal{N}^{-1}(\mathbf{x}_v;\ \boldsymbol{\eta}_v, \boldsymbol{\Lambda}_v) = \prod_{f \in \mathbf{F}_{\mathbf{x}_v}} \mathbf{m}_{f \to \mathbf{x}_v}(\mathbf{x}_v)\ , \tag{2.40}$$

where $\mathbf{m}_{f \to \mathbf{x}_v}(\mathbf{x}_v) = \mathcal{N}^{-1}(\mathbf{x}_v; \boldsymbol{\eta}_{f \to \mathbf{x}_v}, \boldsymbol{\Lambda}_{f \to \mathbf{x}_v})$ is the message from a factor to the variable. As a reminder, the messages are Gaussian beliefs, and in the Information (Canonical) representation of Gaussian distributions, this product can be rewritten as a summation:

$$\boldsymbol{\eta}_v = \sum_{f \in \mathbf{F}_{x_v}} \boldsymbol{\eta}_{f \to \mathbf{x}_v}\ , \qquad \boldsymbol{\Lambda}_v = \sum_{f \in \mathbf{F}_{x_v}} \boldsymbol{\Lambda}_{f \to \mathbf{x}_v}\ . \tag{2.41}$$

2.3.2 Variable to Factor Message

A message from a variable $\mathbf{x}_v$ to a factor $f_k \in \mathbf{F}_{x_v}$ is the product of all incoming factor to variable messages *except* the message from f_k:

$$\mathbf{m}_{\mathbf{x}_v \to f_k}(f_k) = \prod_{f \in \mathbf{F}_{x_v} \setminus f_k} \mathbf{m}_{f \to \mathbf{x}_v}(\mathbf{x}_v)\ . \tag{2.42}$$

2.3.3 Factor to Variable Message

To create a message from a factor f_k to a variable $\mathbf{x}_v \in \mathbf{X}_{f_k}$, we perform three steps. First we calculate the factor potential, then condition the potential on the messages from all variables *except* $\mathbf{x}_v$, and finally marginalise out over those variables.

2.3.3.1 Factor Potential and Linearisation

Before creating the outgoing messages to the factor's connected variables, we calculate the factor potential $\phi(\mathbf{X}_{f_k}; \mathbf{z})$ which is a likelihood function; the probability of obtaining a measurement or observation $\mathbf{z}$ with its associated uncertainty (represented by a precision matrix $\mathbf{\Lambda}$) as a function of the states of the factor's connected variables.

$$\phi(\mathbf{X}_{f_k}; \mathbf{z}) = p(\mathbf{z}|\mathbf{X}_{f_k}) = k_1 \exp\left(-\frac{1}{2}(\mathbf{z} - \boldsymbol{h}(\mathbf{X}_{f_k}))^\top \mathbf{\Lambda}(\mathbf{z} - \boldsymbol{h}(\mathbf{X}_{f_k}))\right) \tag{2.43}$$

$$= k_1 \exp(-C'(\mathbf{X}_{f_k}; \mathbf{z})) \ , \tag{2.44}$$

where k_1 is a constant, and $C'(\mathbf{X}_{f_k}; \mathbf{z})$ can be thought of as a general factor *cost* or *energy*.

In order to use the factor potential in GBP, we must represent it using the information vector $\boldsymbol{\eta}_k$ and precision matrix Λ_k which requires a modified cost function:

$$\phi(\mathbf{X}_{f_k}; \mathbf{z}) = k_2 \exp(-C(\mathbf{X}_{f_k}; \mathbf{z})) \tag{2.45}$$

$$C(\mathbf{X}_{f_k}; \mathbf{z}) = \frac{1}{2}\mathbf{X}_{f_k}^\top \mathbf{\Lambda}_k \mathbf{X}_{f_k} - \boldsymbol{\eta}_k^\top \mathbf{X}_{f_k} \ , \tag{2.46}$$

where k_2 is a constant. Any general linear measurement function can be written as

$$\boldsymbol{h}(\mathbf{X}_{f_k}) = \mathbf{A}\mathbf{X}_{f_k} + \mathbf{b} \ , \tag{2.47}$$

which we can substitute into the original factor cost function $C'(\mathbf{X}_{f_k}; \mathbf{z})$:

$$C'(\mathbf{X}_{f_k}; \mathbf{z}) = \frac{1}{2}((\mathbf{z} - \mathbf{b}) - \mathbf{A}\mathbf{X}_{f_k})^\top \mathbf{\Lambda}((\mathbf{z} - \mathbf{b}) - \mathbf{A}\mathbf{X}_{f_k}) \tag{2.48}$$

$$= \frac{1}{2}\left[(\mathbf{z} - \mathbf{b})^\top \mathbf{\Lambda}(\mathbf{z} - \mathbf{b}) + (\mathbf{A}\mathbf{X}_{f_k})^\top \mathbf{\Lambda}\mathbf{A}\mathbf{X}_{f_k} - 2(\mathbf{z} - \mathbf{b})^\top \mathbf{\Lambda}\mathbf{A}\mathbf{X}_{f_k}\right] \tag{2.49}$$

$$= \frac{1}{2}\mathbf{X}_{f_k}^\top \mathbf{A}^\top \mathbf{\Lambda}\mathbf{A}\mathbf{X}_{f_k} - (\mathbf{z} - \mathbf{b})^\top \mathbf{\Lambda}\mathbf{A}\mathbf{X}_{f_k} + k_3 \tag{2.50}$$

$$= C(\mathbf{X}_{f_k}; \mathbf{z}) + k_3 \ , \tag{2.51}$$

where k_3 is a constant that does not depend on $\mathbf{X}_{f_k}$, and so from equation 2.46:

$$\boldsymbol{\eta}_k = \mathbf{A}^\top \mathbf{\Lambda}(\mathbf{z} - \mathbf{b}) \ , \qquad \mathbf{\Lambda}_k = \mathbf{A}^\top \mathbf{\Lambda}\mathbf{A} \ . \tag{2.52}$$

However, if the measurement function $\boldsymbol{h}(\mathbf{X}_{f_k})$ (and therefore the factor) is *non-linear*, we can simply *choose* a suitable point $\mathbf{X}^t_{f_k}$ to linearise around:

$$\boldsymbol{h}(\mathbf{X}_{f_k}) \approx \boldsymbol{h}(\mathbf{X}^t_{f_k}) + \nabla \boldsymbol{h}(\mathbf{X}_{f_k})\big|_{\mathbf{X}^t_{f_k}} (\mathbf{X}_{f_k} - \mathbf{X}^t_{f_k}) = \mathbf{J}\mathbf{X}_{f_k} + (\boldsymbol{h}(\mathbf{X}^t_{f_k}) - \mathbf{J}\mathbf{X}^t_{f_k}) \ . \tag{2.53}$$

Where $\mathbf{J}$ is the Jacobian of the measurement function evaluated at $\mathbf{X}^t_{f_k}$. Comparing with Equation 2.47 we find that $\mathbf{A} = \mathbf{J}$ and $\mathbf{b} = \boldsymbol{h}(\mathbf{X}^t_{f_k}) - \mathbf{J}\mathbf{X}^t_{f_k}$. Finally, the linearised factor potential $\phi(\mathbf{X}_{f_k}; \mathbf{z})$ can be described by:

$$\boldsymbol{\eta}_k = \mathbf{J}^\top \boldsymbol{\Lambda}(\mathbf{z} - \boldsymbol{h}(\mathbf{X}^t_{f_k}) + \mathbf{J}\mathbf{X}^t_{f_k}) \ , \qquad \boldsymbol{\Lambda}_k = \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} \ . \tag{2.54}$$

2.3.3.2 Conditioning the Factor Potential

Having calculated the linearised factor potential based on the current states of neighbouring variables, we *condition* the potential on the incoming messages from all variables *except* $\mathbf{x}_v$:

$$\phi'_{\mathbf{x}_v}(\mathbf{X}_{f_k}; \mathbf{z}) = \phi(\mathbf{X}_{f_k}; \mathbf{z}) \cdot \prod_{\mathbf{x} \in \mathbf{X}_{f_k} \setminus \mathbf{x}_v} \mathbf{m}_{\mathbf{x} \to f_k}(\mathbf{x}) \ . \tag{2.55}$$

By representing the factor potential in Information form (instead of Moments form) conditioning can be achieved by simply adding the information vectors and precision matrices of the incoming messages to the corresponding slices and diagonal entries of the factor potential. As an example let us consider a factor f_k connected to two variables $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$, and that we are in the process of creating a message to send to variable $\mathbf{x}_\alpha$.

The linearised factor potential parameters are:

$$\boldsymbol{\eta}_k = \begin{bmatrix} \boldsymbol{\eta}_{k_\alpha} \\ \boldsymbol{\eta}_{k_\beta} \end{bmatrix} , \ \boldsymbol{\Lambda}_k = \begin{bmatrix} \boldsymbol{\Lambda}_{k_{\alpha\alpha}} & \boldsymbol{\Lambda}_{k_{\alpha\beta}} \\ \boldsymbol{\Lambda}_{k_{\beta\alpha}} & \boldsymbol{\Lambda}_{k_{\beta\beta}} \end{bmatrix} \ . \tag{2.56}$$

Conditioning the factor potential on the incoming message from variable $\mathbf{x}_\beta$, we get:

$$\boldsymbol{\eta}'_k = \begin{bmatrix} \boldsymbol{\eta}_{k_\alpha} \\ \boldsymbol{\eta}_{k_\beta} + \boldsymbol{\eta}_{\mathbf{x}_\beta \to f_k} \end{bmatrix} , \ \boldsymbol{\Lambda}'_k = \begin{bmatrix} \boldsymbol{\Lambda}_{k_{\alpha\alpha}} & \boldsymbol{\Lambda}_{k_{\alpha\beta}} \\ \boldsymbol{\Lambda}_{k_{\beta\alpha}} & \boldsymbol{\Lambda}_{k_{\beta\beta}} + \boldsymbol{\Lambda}_{\mathbf{x}_\beta \to f_k} \end{bmatrix} \ . \tag{2.57}$$

2.3.3.3 Marginalisation

Finally from the joint distribution described by the conditioned factor potential we marginalise out all variables except the output variable $\mathbf{x}_\alpha$. In general the outgoing message to a variable $\mathbf{x}_v$ is:

$$\mathbf{m}_{f_k \to \mathbf{x}_v}(\mathbf{x}_v) = \sum_{\mathbf{x} \in \mathbf{X}_{f_k} \setminus \mathbf{x}_v} \phi'_{\mathbf{x}_v}(\mathbf{x}; \mathbf{z}) \ . \tag{2.58}$$

A formula for marginalising a general partitioned Gaussian multivariate state in Information form is given in [Eustice et al., 2005]. The outgoing message to variable $\mathbf{x}_\alpha$ is therefore parametrised by:

$$\boldsymbol{\eta}_{f_k \to \mathbf{x}_\alpha} = \boldsymbol{\eta}_{k_\alpha} - \boldsymbol{\Lambda}_{k_{\alpha\beta}}(\boldsymbol{\Lambda}_{k_{\beta\beta}} + \boldsymbol{\Lambda}_{\mathbf{x}_\beta \to f_k})^{-1}(\boldsymbol{\eta}_{k_\beta} + \boldsymbol{\eta}_{\mathbf{x}_\beta \to f_k}) \tag{2.59}$$

$$\boldsymbol{\Lambda}_{f_k \to \mathbf{x}_\alpha} = \boldsymbol{\Lambda}_{k_{\alpha\alpha}} - \boldsymbol{\Lambda}_{k_{\alpha\beta}}(\boldsymbol{\Lambda}_{k_{\beta\beta}} + \boldsymbol{\Lambda}_{\mathbf{x}_\beta \to f_k})^{-1}\boldsymbol{\Lambda}_{k_{\beta\alpha}} \ . \tag{2.60}$$

Note: We made two assumptions in this marginalisation example:

- The factor was connected to only two variables, as is the case for the work in this thesis. In the general case of factors connected to more than two variables, we refer the reader to [Eustice et al., 2005].

- The information vector and precision matrix for the factor potential in Equation 2.56 has been *arranged* such that the output variable appears *first*. If this is not the case (i.e. when creating an outgoing message for $\mathbf{x}_\beta$), $\boldsymbol{\eta}_k$ and $\boldsymbol{\Lambda}_k$ must be reordered.

2.4 Lie Groups for Gaussian Belief Propagation

2.4.1 Motivation and Lie Groups in Robotics

So far, we have assumed that the variables of interest lie in Euclidean (vector) space, allowing beliefs to be represented as Gaussian. This makes inference and optimisation straightforward: gradients are easy to compute, and updates can be applied component-wise in a linear vector space.

However, many problems in robotics involve variables that do not live in vector spaces. Robot orientation, pose, and transformations of coordinate frames are all examples of quantities that lie on non-linear manifolds. For example, a robot's state in 2D is usually represented by its position $(x, y) \in \mathbb{R}^2$ and heading angle $\theta \in \mathrm{S}^1$. Although this state consists of three elements, it does not belong to $\mathbb{R}^3$; rather, it lies on the Lie group SE(2), which jointly captures both translation and rotation in the plane.

This distinction matters during optimisation. In $\mathbb{R}^n$ we compute derivatives by perturbing individual components:

$$\mathbf{X} + \delta \boldsymbol{e}_i = [\mathrm{X}_1, ..., \mathrm{X}_i + \delta, ..., \mathrm{X}_n]^\top \ . \tag{2.61}$$

But consider optimising a function that operates on a 2D rotation matrix. These 2×2 matrices are constrained to be orthogonal with a determinant of one. Naively perturbing an element of the matrix violates these constraints, resulting in an invalid rotation. A common alternative is to parametrise rotations using Euler angles, but this representation suffers from singularities (e.g., gimbal lock), where small changes in orientation can cause large, discontinuous jumps in the parameters, making optimisation unstable.

Lie groups provide the right framework to perform valid updates that respect the underlying geometry. They are smooth manifolds equipped with a group operation, and their associated tangent spaces allow for linearisation. This structure allows us to perform in-

ference and optimisation using linear tools such as Jacobians and Gaussian distributions.

Lie theory is widely used in robotics, particularly in state estimation, SLAM, and motion planning [Drummond and Cipolla, 2000], [Solà, 2017], [Mukadam et al., 2018a]. Working on manifolds such as SO(2), SE(2), or SE(3) allows us to represent robot configurations naturally while preserving the group structure.

We now introduce the key ideas from Lie theory required for inference on manifolds, and to generalise GBP to variables represented as members of Lie groups.

2.4.2 Lie Groups and their Tangent Spaces

Lie theory is named after the mathematician Sophus Lie (1842 – 1899), and has been extensively studied. Although the theory can be very complex, its use in robotics requires only part of that deep theory. We refer the reader to "A Micro Lie Theory for State Estimation in Robotics" [Deray and Solà, 2020] for further details, although we present a minimal introduction in this chapter.

A Lie group is a set that takes on the properties of both a mathematical group as well as a smooth manifold. A group is a set equipped with a binary operation known as 'composition' which combines two elements of the set to produce another. A set $\mathcal{M}$ is a group under a composition operator $\circ$ if the following properties hold:

- **Closure:** For all $\mathcal{X}, \mathcal{Y} \in \mathcal{M}$, the composition $\mathcal{X} \circ \mathcal{Y} \in \mathcal{M}$,
- **Associativity:** For all $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathcal{M}$, $(\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z})$,
- **Identity** and **Inverse:** For all $\mathcal{X} \in \mathcal{M}$ there exists an **identity** element $\mathcal{E} \in \mathcal{M}$ such that $\mathcal{X} \circ \mathcal{E} = \mathcal{E} \circ \mathcal{X}$, and an **inverse** element such that $\mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{E}$.

A Lie group is also a **smooth manifold** which is a continuous, differentiable space that

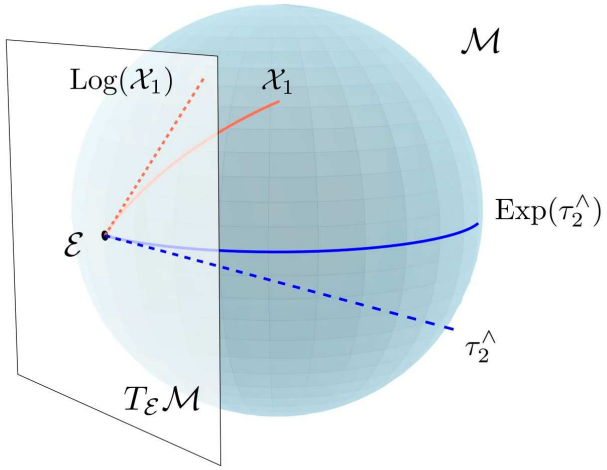


Figure 2.4: The manifold (visualised as a blue sphere) for a Lie group $\mathcal{M}$, along with its Lie algebra $T_\mathcal{E}\mathcal{M}$ (the tangent space at the Identity element $\mathcal{E}$. The $\text{Log}(\cdot)$ map transforms a point on the manifold $\mathcal{X}$ onto $T_\mathcal{E}\mathcal{M}$, and the $\text{Exp}(\cdot)$ map retracts a vector $\tau^\wedge$ from $T_\mathcal{E}\mathcal{M}$ back onto the manifold.

looks Euclidean (linear) at a local level. This means that we can perform calculus on the group, which is needed for optimisation in robotics and estimation.

The smoothness of the manifold means that there exists a **tangent space** at each point on the manifold (at each element of the Lie group). The Lie algebra $\mathfrak{m}$ of a Lie group $\mathcal{M}$ is the name given to the tangent space at the Identity element $T_\mathcal{E}\mathcal{M}$. This is a local, linear space where we can perform standard vector operations, which we cannot directly perform on elements of the Lie group. Although this space is not exactly the same as Euclidean space, it is *isomorphic* to $\mathbb{R}^m$ for some m; it is often simpler to perform vector operations on this space rather than on $T_\mathcal{E}\mathcal{M}$. We define the *hat* and *vee* operators to move between $\mathbb{R}^m$ and $\mathcal{M}$:

$$\text{Hat: } (\cdot)^\wedge : \mathbb{R}^m \to \mathfrak{m} \,, \qquad \text{Vee: } (\cdot)^\vee : \mathfrak{m} \to \mathbb{R}^m \,. \tag{2.62}$$

To move between the Lie group $\mathcal{M}$ and its Lie algebra $\mathfrak{m}$ we use two maps:

- The **exponential map** $\exp : \mathfrak{m} \to \mathcal{M}$, which maps a vector in the tangent space

to a point on the manifold.

- The **logarithmic map** $\log : \mathcal{M} \to \mathfrak{m}$, which produces the projection of the Lie group element in the tangent space.

These maps are crucial to working with Lie groups, as they let us apply small perturbations in the tangent space, and project the result back onto the Lie group manifold, enabling optimisation and propagation of uncertainty on the manifold.

As a shortcut we can define the *capitalised* Exp and Log maps which directly relate the manifold $\mathcal{M}$ and the Euclidean $\mathbb{R}^m$ space:

$$\text{Exp: } \mathbb{R}^m \underset{(\cdot)^\wedge,\ \exp}{\to} \mathcal{M}\ , \qquad \text{Log: } \mathcal{M} \underset{\log,\ (\cdot)^\vee}{\to} \mathbb{R}^m\ . \tag{2.63}$$

These operations are shown visually in Figure 2.4.

As a concrete example let us consider the Lie group SO(2) which is the group of 2D rotations. Each element $\mathcal{X} \in \mathrm{SO}(2)$ can be represented by a 2×2 orthogonal matrix with determinant 1, parametrised by a rotation angle $\theta \in \mathbb{R}^1$.

The Lie algebra $\mathfrak{so}(2)$ consists of all 2×2 skew-symmetric matrices. These matrices are over-parametrised and contain only one degree of freedom; we can represent them more compactly as $\theta \in \mathbb{R}^1$. For this Lie group, the *hat* operator maps a scalar value θ into a skew-symmetric 2×2 matrix $\theta^\wedge = [\theta]_\times \in \mathfrak{so}(2)$, and the *vee* performs the opposite. The Exponential and Logarithmic maps are:

$$\mathrm{Exp}(\theta) = \exp(\theta^\wedge) = \exp\left(\begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}\right) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \mathcal{X} \tag{2.64}$$

$$\mathrm{Log}(\mathcal{X}) = \theta = \mathrm{atan2}(\mathcal{X}_{21}, \mathcal{X}_{11}) \tag{2.65}$$

2.4.2.1 On-manifold Operations

To work with Lie groups, we define addition and subtraction using the manifold's tangent space. Given a point $\mathcal{X} \in \mathcal{M}$, we can move to a new point $\mathcal{Y}$ by 'adding' a tangent vector $\boldsymbol{\tau} \in \mathbb{R}^m$ using the Exponential map, and the relative displacement from $\mathcal{X}$ to $\mathcal{Y}$ (in the tangent space to $\mathcal{X}$) is computed using the Logarithmic map:

$$\mathcal{Y} = \mathcal{X} \oplus \boldsymbol{\tau} \triangleq \mathcal{X} \circ \mathrm{Exp}(\boldsymbol{\tau}) \in \mathcal{M} \ , \tag{2.66}$$

$$\boldsymbol{\tau} = \mathcal{Y} \ominus \mathcal{X} \triangleq \mathrm{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \in T_{\mathcal{X}}\mathcal{M} \ . \tag{2.67}$$

where the $\oplus$ and $\ominus$ operations mimic vector addition and subtraction, enabling local computations on manifolds via 'retraction' to and from the tangent space.

2.4.2.2 Lie Group Jacobians

Recall the standard definition of a derivative in Euclidean space is:

$$\frac{\mathrm{d}f(\mathbf{x})}{\mathrm{d}\mathbf{x}} = \lim_{\mathbf{x}\to 0} \frac{f(\mathbf{x}+\delta\mathbf{x}) - f(\mathbf{x})}{\delta\mathbf{x}} \ , \tag{2.68}$$

where the function $f : \mathbb{R}^m \to \mathbb{R}^n$ is multivariate, and the resulting Jacobian matrix is an $n \times m$ matrix. To use this definition of derivatives with functions operating on Lie groups $f : \mathcal{M} \to \mathcal{N}$, we can simply use the $\oplus$ and $\ominus$ operators defined previously to obtain:

$$\frac{\mathcal{D}f(\mathcal{X})}{\mathcal{D}\mathcal{X}} = \lim_{\delta\boldsymbol{\tau}\to 0} \frac{f(\mathcal{X} \oplus \delta\boldsymbol{\tau}) \ominus f(\mathcal{X})}{\delta\boldsymbol{\tau}} \ , \tag{2.69}$$

where $\delta\boldsymbol{\tau} \in T_{\mathcal{E}}\mathcal{M}$. This generalises the Jacobian to Lie groups by expressing derivatives with respect to tangent space perturbations.

2.4.3 Uncertainty on the Manifold

We follow the convention of [Deray and Solà, 2020] and define a random variable over a Lie group $\mathcal{M}$ with a right-perturbation. Without loss of generality let us consider a

random variable $\mathcal{X} \in \mathrm{SO}(2)$, which represents a 2D rotation matrix:

$$\mathcal{X} = \bar{\mathcal{X}} \oplus \boldsymbol{\xi} \ , \tag{2.70}$$

where $\bar{\mathcal{X}} \in \mathrm{SO}(2)$ is the mean rotation of the random variable, and $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{X}}) \in \mathfrak{so}(2)$ is a small Gaussian-distributed perturbation in the tangent space around $\bar{\mathcal{X}}$ with covariance matrix $\Sigma_{\mathcal{X}} \in \mathbb{R}^{m \times m}$. We will use the notation $\mathcal{X} \sim \mathcal{N}(\bar{\mathcal{X}}, \boldsymbol{\Sigma}_{\mathcal{X}})$ to denote a random variable $\mathcal{X}$ on the manifold.

We follow the process in [Murai et al., 2024b] to use Lie theory within GBP:

> **All messages to and from a Lie group variable are point estimates $\mathcal{X}$, along with a precision (inverse covariance) matrix $\boldsymbol{\Lambda}$ defined in the tangent space around that element.**

We now outline the modifications to the steps of GBP when representing variables and factors that use Lie theory.

2.4.4 Variable Belief Update

2.4.4.1 Transformation of Incoming Messages

Each message from a factor $f_k \in \mathbf{F}_{\mathcal{X}_v}$ connected to a variable $\mathcal{X}_v$ represents that factor's belief about it; the message $\mathbf{m}_{f_k \to \mathcal{X}_v} = \mathcal{N}(\bar{\mathcal{X}}_{f_k \to \mathcal{X}_v},\ \boldsymbol{\Lambda}^{-1}_{f_k \to \mathcal{X}_v})$ consists of a point $\bar{\mathcal{X}}_{f_k \to \mathcal{X}_v}$ on the manifold and a precision matrix defined in the tangent space $T_{\bar{\mathcal{X}}_{f_k \to \mathcal{X}_v}} \mathcal{M}$ at that point. In order to combine all factor-to-variable messages, they must first be transformed onto a common tangent space, such as the space $T_{\bar{\mathcal{X}}_{v,t-1}} \mathcal{M}$ at the mean $\bar{\mathcal{X}}_{v,t-1}$ of the variable's previously calculated belief. This process is visualised in Figure 2.5.

First, the means of the incoming messages are transformed into tangent vectors and

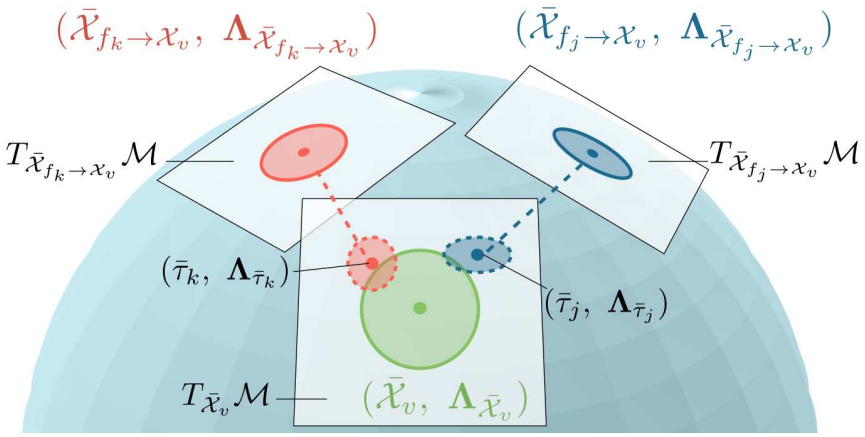


Figure 2.5: The messages from factors f_j and f_k to a variable $\mathcal{X}_v$ are shown respectively as the solid red and blue ellipses on the tangent spaces at their means. The dotted ellipses represent the same messages *transformed* onto the tangent space at variable $\mathcal{X}_v$'s current belief mean (green).

precision matrices on $T_{\bar{\mathcal{X}}_{v,t-1}}\mathcal{M}$. For the message from factor f_k we have:

$$\bar{\boldsymbol{\tau}}_k = \bar{\mathcal{X}}_{f_k \to \mathcal{X}_v} \ominus \bar{\mathcal{X}}_{v,t-1} \ \in \mathcal{T}_{\bar{\mathcal{X}}_{v,t-1}}\mathcal{M} \ , \tag{2.71}$$

$$\mathbf{\Lambda}_{\bar{\boldsymbol{\tau}}_k} = \boldsymbol{J}(\bar{\boldsymbol{\tau}}_k)^\top \mathbf{\Lambda}_{f_k \to \mathcal{X}_v} \boldsymbol{J}(\bar{\boldsymbol{\tau}}_k) \ , \tag{2.72}$$

where $\boldsymbol{J}(\boldsymbol{\tau})$ is the right Jacobian of $\mathcal{X} = \mathrm{Exp}(\boldsymbol{\tau})$:

$$\boldsymbol{J}(\boldsymbol{\tau}) = \frac{\mathcal{D}\ \mathrm{Exp}(\boldsymbol{\tau})}{\mathcal{D}\boldsymbol{\tau}} = \lim_{\delta \to 0} \frac{\mathrm{Exp}(\boldsymbol{\tau} + \delta) \ominus \mathrm{Exp}(\boldsymbol{\tau})}{\delta} \ . \tag{2.73}$$

2.4.4.2 Computation of Incremental Belief and Variable State Update

Having warped all incoming messages onto a common tangent space, we can proceed with the belief update steps to compute the new marginal belief for the variable. **Note:** this new belief is also in the tangent plane of our existing belief mean $\bar{\mathcal{X}}_{v,t-1}$, and so it can be considered to be an 'incremental belief'.

We form the new incremental belief mean $\boldsymbol{\tau}_v$ by calculating its information $\boldsymbol{\eta}_{\boldsymbol{\tau}_v}$ and

precision $\boldsymbol{\Lambda}_{\boldsymbol{\tau}_v}$:

$$\boldsymbol{\eta}_{\boldsymbol{\tau}_v} = \sum_{f_k \in \mathbf{F}_{\mathcal{X}_v}} \boldsymbol{\Lambda}_{\bar{\boldsymbol{\tau}}_k} \bar{\boldsymbol{\tau}}_k \ , \qquad \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v} = \sum_{f_k \in \mathbf{F}_{\mathcal{X}_v}} \boldsymbol{\Lambda}_{\bar{\boldsymbol{\tau}}_k} \ , \tag{2.74}$$

$$\boldsymbol{\tau}_v = \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v}^{-1} \boldsymbol{\eta}_{\boldsymbol{\tau}_v} \ . \tag{2.75}$$

Finally we retract this incremental belief mean onto the manifold and use it to update our previous belief mean $\bar{\mathcal{X}}_{v,t}$ and precision $\boldsymbol{\Lambda}_{\bar{\mathcal{X}}_{v,t}}$:

$$\bar{\mathcal{X}}_{v,t} = \bar{\mathcal{X}}_{v,t-1} \oplus \boldsymbol{\tau}_v \ , \qquad \boldsymbol{\Lambda}_{\bar{\mathcal{X}}_{v,t}} = \boldsymbol{J}(\boldsymbol{\tau}_v)^{-\top} \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v} \boldsymbol{J}(\boldsymbol{\tau}_v)^{-1} \ . \tag{2.76}$$

2.4.5 Variable to Factor Message

The message $\mathbf{m}_{\mathcal{X}_v \to f_k}$ from variable $\mathcal{X}_v$ to a factor f_k is the product of *all* messages to the variable *except* that from factor f_k. From the incremental belief calculated in the previous step, we filter out the message from f_k to get a new outgoing information vector, precision matrix and incremental belief mean $\boldsymbol{\tau}_{\mathcal{X} \to f_k}$:

$$\boldsymbol{\eta}_{\boldsymbol{\tau}_v \to f_k} = \boldsymbol{\eta}_{\boldsymbol{\tau}_v} - \boldsymbol{\Lambda}_{\bar{\boldsymbol{\tau}}_k} \bar{\boldsymbol{\tau}}_k \ , \qquad \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v \to f_k} = \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v} - \boldsymbol{\Lambda}_{\bar{\boldsymbol{\tau}}_k} \ , \tag{2.77}$$

$$\boldsymbol{\tau}_{\mathcal{X} \to f_k} = \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v \to f_k}^{-1} \, \boldsymbol{\eta}_{\boldsymbol{\tau}_v \to f_k} \tag{2.78}$$

Note that we must first **recalculate** Equations 2.71 and 2.72 using the new belief $\mathcal{X}_{v,t}$. Finally the outgoing message $\mathbf{m}_{\mathcal{X}_v \to f_k} = \mathcal{N}(\bar{\mathcal{X}}_{\mathcal{X}_v \to f_k}, \ \Lambda_{\mathcal{X}_v \to f_k}^{-1})$ is calculated by retracting the incremental belief mean onto the current belief of the variable:

$$\bar{\mathcal{X}}_{\mathcal{X}_v \to f_k} = \bar{\mathcal{X}}_{v,t} \oplus \boldsymbol{\tau}_{\mathcal{X} \to f_k} \tag{2.79}$$

$$\boldsymbol{\Lambda}_{\mathcal{X}_v \to f_k}^{-1} = \boldsymbol{J}(\boldsymbol{\tau}_{\mathcal{X} \to f_k})^{-\top} \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v \to f_k} \boldsymbol{J}(\boldsymbol{\tau}_{\mathcal{X} \to f_k})^{-1} \ . \tag{2.80}$$

2.4.6 Factor to Variable Message

Let us consider a factor f_k sending a message to each of its connected variables $\mathcal{X}_v \in \mathbf{X}_{f_k}$. The message is the product of the linearised factor potential $\phi(\boldsymbol{\tau}_a; \boldsymbol{\eta}_a, \boldsymbol{\Lambda}_a)$, with all

incoming messages *except* the one from variable $\mathcal{X}_v$, marginalised over all variables except $\mathcal{X}_v$.

2.4.6.1 Warping of Incoming Messages

As in the previous sections, we must first transform all incoming messages into a common tangent plane: the one at the factor's current linearisation point $\bar{\mathcal{X}}_{k,t}$:

$$\bar{\boldsymbol{\tau}}_v = \bar{\mathcal{X}}_{\mathcal{X}_v \to f_k} \ominus \bar{\mathcal{X}}_{k,t} \ , \qquad \boldsymbol{\Lambda}_{\boldsymbol{\tau}_v} = \boldsymbol{J}(\bar{\boldsymbol{\tau}}_v)^\top \boldsymbol{\Lambda}_{\mathcal{X}_v \to f_k} \boldsymbol{J}(\bar{\boldsymbol{\tau}}_v) \ . \tag{2.81}$$

2.4.6.2 Linearisation and Factor Potential Calculation

The constraint represented by a factor can be represented by a measurement function $\boldsymbol{h}(\mathcal{X})$ which is a function of one or more Lie group elements $\mathcal{X} \in \mathcal{M}$. As in Section 2.3.3.1, we can perform a Taylor series expansion over $\boldsymbol{h}$, by perturbing a point $\bar{\mathcal{X}}_{k,t}$ by a small vector $\bar{\boldsymbol{\tau}} = \mathcal{X} \ominus \bar{\mathcal{X}}_{k,t} \in \mathcal{T}_{\mathcal{X}}\mathcal{M}$ in the tangent space at $\bar{\mathcal{X}}_{k,t}$:

$$\boldsymbol{h}(\mathcal{X}) \approx \boldsymbol{h}(\bar{\mathcal{X}}_{k,t}) + \mathbf{J}\bar{\boldsymbol{\tau}} \ , \tag{2.82}$$

where $\mathbf{J} = \nabla \boldsymbol{h}(\mathcal{X})\big|^\top_{\bar{\mathcal{X}}_{k,t}}$ is the Jacobian of the measurement function evaluated at $\bar{\mathcal{X}}_{k,t}$. Note that this linearised function is a function of $\boldsymbol{\tau} \in T_{\bar{\mathcal{X}}_{k,t}}\mathcal{M}$ which is in the tangent space at $\mathcal{X}_{k,t}$.

Comparing Equation 2.82 with Equation 2.47 we get $\mathbf{A} = \mathbf{J}$ and $\mathbf{b} = \boldsymbol{h}(\mathcal{X}_{k,t})$. This leads us to the factor potential $\phi(\boldsymbol{\tau}; \boldsymbol{\eta}_k, \boldsymbol{\Lambda}_k)$ which is a function of $\boldsymbol{\tau}$ given by:

$$\boldsymbol{\eta}_k = \mathbf{J}^\top \boldsymbol{\Lambda}(\mathbf{z} - \boldsymbol{h}(\mathcal{X}_t)) \ , \qquad \boldsymbol{\Lambda}_k = \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} \ . \tag{2.83}$$

2.4.6.3 Conditioning and Marginalisation

Conditioning and Marginalisation of the factor potential takes place in the same way as in Sections 2.3.3.2 and 2.3.3.3, and results in the incremental marginal belief of the

receiving variable $\mathcal{X}_v$ parametrised by $\boldsymbol{\eta}_{f_k \to \tau_v}$ and $\boldsymbol{\Lambda}_{f_k \to \tau_v}$. Finally, the incremental belief is retracted to create the outgoing message $\mathbf{m}_{f_k \to \mathcal{X}_v} = \mathcal{N}(\bar{\mathcal{X}}_{f_k \to \mathcal{X}_v},\ \boldsymbol{\Lambda}^{-1}_{f_k \to \mathcal{X}_v})$:

$$\bar{\boldsymbol{\tau}}_{f_k \to \mathcal{X}_v} = \boldsymbol{\Lambda}^{-1}_{f_k \to \mathcal{X}_v} \boldsymbol{\eta}_{f_k \to \mathcal{X}_v} \ , \tag{2.84}$$

$$\bar{\mathcal{X}}_{f_k \to \mathcal{X}_v} = \bar{\mathcal{X}}_{v,t} \oplus \bar{\boldsymbol{\tau}}_{f_k \to \mathcal{X}_v} \ , \tag{2.85}$$

$$\boldsymbol{\Lambda}_{f_k \to \mathcal{X}_v} = \mathbf{J}(\bar{\boldsymbol{\tau}}_{f_k \to \mathcal{X}_v})^{-\top} \boldsymbol{\Lambda}_{\bar{\boldsymbol{\tau}}_{f_k \to \mathcal{X}_v}} \mathbf{J}(\bar{\boldsymbol{\tau}}_{f_k \to \mathcal{X}_v})^{-1} \ . \tag{2.86}$$

2.5 Summary

In this chapter we have introduced factor graphs as a form of probabilistic graphical models for solving optimisation problems with sparsely connected variables.

We have formally introduced Gaussian Belief Propagation (GBP) as a distributed algorithm for inference over factor graph, used throughout the work in this thesis. Chapters 3 and 4 use the standard (Euclidean) version of GBP as described in Section 2.3. Chapters 5 and 6 make use of GBP with Lie theory as described in Section 2.4.

For a more detailed discussion of GBP, the reader is referred to the work in [Davison and Ortiz, 2019], and the accompanying interactive article [Ortiz et al., 2021].

Chapter 3

GBP Planner: Distributing Collaborative Multi-Robot Planning With Gaussian Belief Propagation

If everyone is moving forward together, then success takes care of itself.

— Henry Ford

Contents

The previous chapter established the foundations of Gaussian Belief Propagation (GBP). We now extend these ideas to a key problem in robotics: distributed multi-robot planning under uncertainty. This chapter presents our first applied contribution, in which robots employ GBP to jointly optimise their trajectories in real time while avoiding collisions.

3.1 Introduction

As automation increases, multiple robotic devices will need to operate together, whether working robots in a home, farm or industrial setting, or autonomous vehicles on a road network. To achieve both safety and efficiency when speeds are high and space is limited, these robots must *coordinate* when planning their motions and other actions. At the most basic level, robots should take minimal account of each other's plans in order to avoid collisions. When coordination is stronger, extremely high efficiency is possible, and large numbers of devices can move smoothly and intricately around each other as they carry out their tasks.

It might be assumed that highly coordinated planning requires centralised control, where a single computer receives state information from all robots and solves a unified multiple trajectory optimisation problem before sending commands back to them all. In this chapter we show that precise performance is in fact possible with an alternative distributed technique which uses efficient local per-robot computation and purely peer-to-peer communication.

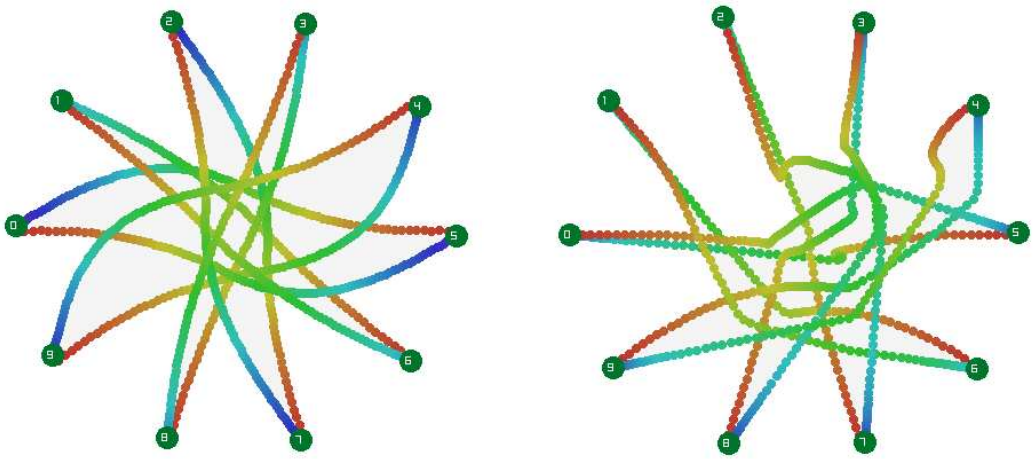


Figure 3.1: Experiment where 10 robots cross to opposite sides of a circle without colliding. Our GBP planner (left) achieves shorter and smoother trajectories than ORCA (right). Colours change along the paths with time, from red to blue.

Our solution formulates multi-robot planning as a general dynamic optimisation problem characterised by a factor graph with variables representing robot positions and velocities in Euclidean space over a bounded forward time window. These variables are connected by constraint factors which represent each robot's dynamics as well as the need to avoid collisions with other robots and static obstacles. Collaborative planning consists of performing inference on the factor graph to determine marginal distributions for the future poses which best satisfy all constraints.

The key novelty of our approach is to show that the inference required for planning within this general factor graph framework can be achieved via distributed computation. We di-

vide the full joint factor graph into fragments which individual robots have responsibility for storing and updating. Inference is then achieved using Gaussian Belief Propagation (GBP), an iterative and distributed message passing algorithm which can achieve the same global solution as a centralised solver. Messages which need to be sent between the graph fragments held by different robots are transmitted by regular peer-to-peer communication of small messages. The particularly appealing property of GBP for multi-robot systems is that global convergence can be achieved with an ad-hoc communication schedule, as would often be the case in real scenarios when for instance robots only achieve intermittent radio contact.

Our GBP Planning algorithm is generic, but we demonstrate it here in simulations of two scenarios of multi-robot planning in tight spaces. In the first, robots must simultaneously exchange places across a circle formation to move from one fixed configuration to another. In the second, two continuous streams of robots meet at a busy junction and aim to cross each other while maintaining high flow rates. In both, the robots jointly plan and update their trajectories online to achieve smooth (dynamically realistic) and efficient trajectories, and we quantitatively measure performance significantly beyond that demonstrated by alternative distributed planning techniques. We also show that the planning performance only degrades slowly when peer-to-peer communication is reduced or unreliable with a large fraction of messages lost.

In summary, our key contributions are:

- The first use of Gaussian Belief Propagation as a general method for distributed multi-robot planning with multiple dynamics and obstacle constraints over a forward window. This leads to a simple and highly generic technique.
- Quantitative and qualitative simulation experiments of 2D multi-robot scenarios that show that the intricate multi-robot coordination achieved by GBP Planning significantly outperforms a well known alternative distributed planning method in this setting.

3.2 Related Work

In single-robot planning, a robot solves for a path towards its goal while balancing the need to move in accordance with dynamics limits and to avoid obstacles. These various aspects are expressed as hard or soft constraints in a cost function, and an algorithm then solves for the trajectory with the minimum total cost. Is often computationally intractable to find the global minimum cost trajectory for complex problems, but iterative solvers can find locally optimal solutions for a finite number of look-ahead steps.

Multi-robot planning is a generalisation of this, where simultaneous trajectory solutions are sought for multiple robots. Multi-robot planning in tight spaces entails conflict resolution because each robot's future planned positions become obstacles for every other robot. Centralised approaches (e.g. [Soria et al., 2021]) can achieve high performance, but require all robots to be coordinated by a single processor; this represents clear challenges to scalability. We therefore focus here on distributed methods like ours, which can operate with per-robot parallelism and peer-to-peer communication.

One possible approach is priority-based planning such as in [van den Berg and Overmars, 2005]. Here, high priority robots largely use single-robot planning methods. Lower-priority robots then treat them as dynamic obstacles with unchanging trajectories which must be avoided, and will therefore often have to wait before being able to reach their goals. While this makes sense in some applications, there are many other situations where there is no meaningful priority ordering. Here conflict resolution becomes more subtle because every change to one robot's plan could affect all the others, and the rest of the methods we discuss here apply to this case.

A widely used distributed multi-robot planner is the Optimal Reciprocal Collision Avoidance (ORCA) algorithm [van den Berg et al., 2009]. In ORCA, every robot has perfect instantaneous knowledge of the position and velocities of its neighbours. Robots react accordingly by changing their own velocities, but cannot make decisions to resolve conflicts

which may arise further into the future. ORCA is fast and has been shown to be effective in some real applications. However, as we will show in comparative results later, ORCA produces trajectories which are often jerky and inefficient when many robots crowd into tight spaces. A related method where all robots treat each other as velocity obstacles but considering more general kinodynamic robot motions was presented in [Alonso-Mora et al., 2018a]. The distributed version of this method only allowed a short look ahead and was demonstrated with very small numbers of robots. Another interesting distributed method, but also without general communication for resolution of multi-robot forward plans, was presented in [Senbaslar et al., 2019] and was demonstrated on six real differential drive robots.

Several existing methods have used iterative communication to enable longer look-ahead. The work in [Van Parys and Pipeleers, 2016] presents an online solver using Alternating Direction Method of Multipliers (ADMM) for a small number of robots moving through an environment with moving obstacles. Here the robots move together in formation and their paths do not intersect so few conflicts need to be resolved. The distributed model predictive control algorithm in [Luis et al., 2020] has been demonstrated for impressive criss-crossing behaviour in teams of real quadrotor robots, though again the gaps between robots are generally larger than we are able to achieve with our approach. The graph neural network method in [Li et al., 2020a] tackles learned multi-robot planning with local communication, though so far only in a very abstract grid-world setting.

In our work we show for the first time a way to achieve a solution to distributed planning using a general multi-robot cost function over an arbitrary length forward time window. We are inspired in particular by methods like GPMP2 (the Gaussian Process Motion Planner) [Mukadam et al., 2018b]. This is a planner for a single robot in a known static environment, but importantly showed that planning with dynamics can commonly be formulated using a cost function with a sum of quadratic terms, and therefore that solving for a trajectory is equivalent to performing inference on a Gaussian factor graph. This means that general and powerful factor graph solvers such as GTSAM (Georgia Tech

Smoothing and Mapping) [Dellaert, 2021] which are normally used for estimation could be easily adapted to planning applications. The definition of all cost terms in a planning cost function as quadratic in a set of variables means that planning is solved by the same sparse least-squares inference computation as arises in robot estimation problems such as SLAM.

Very recently, Murai *et al.* showed in their Robot Web work [Murai et al., 2024b] that a highly distributed solution to the distributed estimation problem of multi-robot localisation is possible using efficient distributed computation via Gaussian Belief Propagation and peer-to-peer message passing. In our work, we adapt this method for the first time to multi-robot *planning*, and show that it is possible to take advantage of the same distributed computation structure. We will show that this enables highly coordinated multi-robot motion planning performance while remaining very flexible and general.

We would like to make clear that expressing a multi-robot planning cost function as a sum of quadratic terms is a highly general formulation. The relative strengths of the terms can all be set as required such that some constraints are much more necessary to obey than others (especially collision avoidance). The quadratic terms can also include non-linear functions in variable space. While this formulation does not strictly allow 'hard' constraints, it allows arbitrarily strong ones, and this is true of other distributed planning algorithms such as [Luis et al., 2020] when used in practice.

3.3 Theoretical Background

3.3.1 Gaussian Belief Propagation (GBP)

In this work we use GBP to perform inference over a factor graph, which models multi-robot planning as illustrated in figure 3.2. We will explain the details of the factors shown in Section 3.4. The reader may refer to Chapter 2 for a formal introduction to

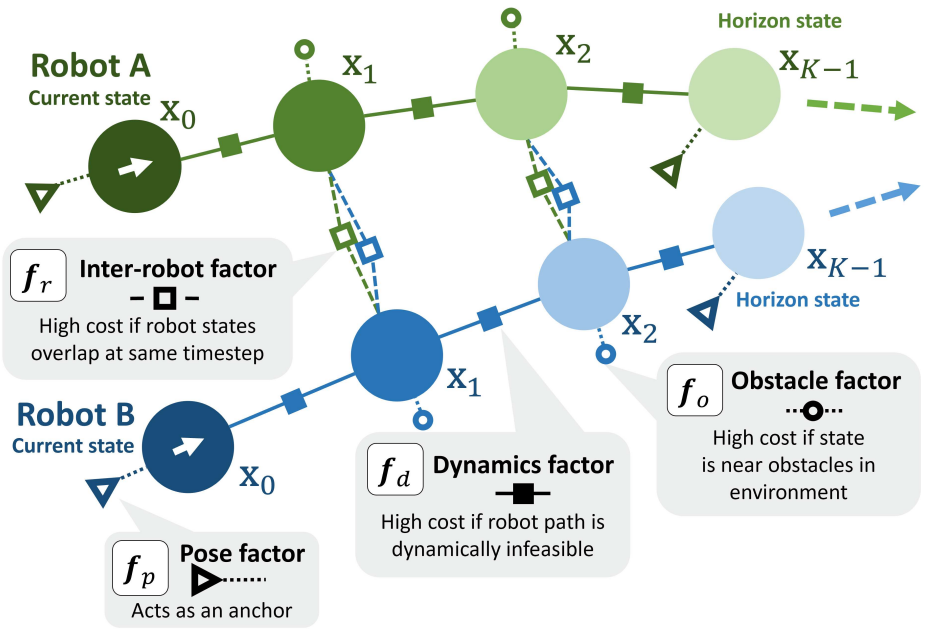


Figure 3.2: Multi-robot planning factor graph with two robots (A and B) moving from bottom-left to top-right highlighting pose (f_p), dynamics (f_d), inter-robot (f_r), and static obstacle (f_o) factors.

factor graphs and GBP. In multi-robot systems, the recent Robot Web approach [Murai et al., 2024b] showed how GBP could be used for decentralised and scalable localisation of groups of up to 1000 robots making noisy inter-robot measurements. In this chapter, we now show for the first time that thanks to the common mathematical formulation of factor graphs, GBP can alternatively be used for multi-robot *planning*.

3.4 Method

We now explain the details of how we formulate multi-robot planning as a factor graph. Although GBP can optimise factor graphs for any type of Gaussian variables and factors, for the rest of the paper we will consider a 2D scenario where robots are modelled as objects with two degrees of movement freedom on a plane.

The state $\mathbf{x}_k$ of a robot at time t_k represents the robot's position and velocity at that

particular moment in time:

$$\mathbf{x}_k = [\boldsymbol{x}_k^\top, \dot{\boldsymbol{x}}_k^\top]^\top = [x_k, y_k, \dot{x}_k, \dot{y}_k]^\top \ . \tag{3.1}$$

The short-term planned trajectory $\mathbf{X}$ of a robot with a lookahead horizon of t_{K-1} seconds is parametrised by K such states at discrete times t_k into the future, from the current state $\mathbf{x}_0$ of the robot at the current time t_0 to the horizon state $\mathbf{x}_{K-1}$ at time t_{K-1}. Each state is a variable in a factor graph and is connected to the previous state in time with a factor representing dynamics. The states are therefore all Markovian in time, resulting in a sparse linear system. The optimal solution for the planned trajectory (the poses from the current state until the lookahead horizon) can be found by solving for the maximum a posteriori (MAP) solution $\mathbf{X}^*$ over the trajectory states. Our method also determines marginal covariances for these planned states. A high covariance will arise in wide open areas where many solutions have similar cost, and a small one where precise movement is needed to satisfy constraints.

In the presence of moving obstacles and other robots the safety of the robot (collision avoidance) is paramount. It is the states in the immediate future that are of more importance when optimising a trajectory. The time gaps between consecutive states therefore increase for states that are further along the planned trajectory.

There are four types of factors that we consider. A unary pose factor f_p is connected to the current and horizon states of the robot and represents the prior information on those states. A dynamics factor f_d encourages a smooth trajectory and represents a noise-on-acceleration dynamics model. An obstacle factor f_o penalises a state from being close to a stationary obstacle in the scene. Finally, an inter-robot factor f_r penalises trajectories that bring two robots within collision range at a particular time step, and takes into account each robot's future positions. An example of the resulting factor graph may be seen in figure 3.2.

Following the convention in Chapter 2, we define a factor f with a measurement function $\boldsymbol{h}(\cdot)$ acting on a set of variables $\mathbf{x}$ connected to the factor. The measurement function

along with an observation $\mathbf{z}$ is assumed to be Gaussian with a precision matrix $\mathbf{\Lambda}$.

3.4.1 Pose Factor

We define a unary pose factor f_p connected to the current and horizon states of the robot with measurement function and precision matrix:

$$\boldsymbol{h}_p(\mathbf{x}_k) = \mathbf{x}_k \ , \tag{3.2}$$

$$\mathbf{\Lambda}_p = \sigma_p^{-2}\mathbf{I} \ . \tag{3.3}$$

The observation $\mathbf{z}_k$ is a constant set to the desired anchoring value of the state $\mathbf{x}_k^0$. The robot current and horizon states ($\mathbf{x}_0$ and $\mathbf{x}_{K-1}$) are connected to strong pose factors with small values of σ_p, ensuring that the trajectory is anchored at these states during optimisation at a time step. At the next time step, these pose factors are modified to reflect the fact that the robot has moved (changed its current state); the observation $\mathbf{z}$ is updated according to a simple dynamics model:

$$\mathbf{z}_0 \rightarrow \boldsymbol{x}_0 + \dot{\boldsymbol{x}}_0 \Delta t \tag{3.4}$$

$$\mathbf{z}_{K-1} \rightarrow \boldsymbol{x}_{K-1} + \dot{\boldsymbol{x}}_{K-1} \Delta t \tag{3.5}$$

3.4.2 Dynamics Factor

For a trajectory to be smooth and dynamically feasible, each state of the robot is connected to the next in time via a factor f_d defined in [Mukadam et al., 2018b] as:

$$\boldsymbol{h}_d(\mathbf{x}_k, \mathbf{x}_{k+1}) = \mathbf{\Phi}(t_{k+1}, t_k)\mathbf{x}_k - \mathbf{x}_{k+1} \ , \tag{3.6}$$

$$\mathbf{\Lambda}_d = \begin{bmatrix} \frac{1}{3}\Delta t_k^3 \mathbf{Q}_d & \frac{1}{2}\Delta t_k^2 \mathbf{Q}_d \\ \frac{1}{2}\Delta t_k^2 \mathbf{Q}_d & \Delta t_k \mathbf{Q}_d \end{bmatrix}^{-1} , \tag{3.7}$$

with $\Delta t_k = t_{k+1} - t_k$, $\mathbf{Q}_d = \sigma_d^2 \mathbf{I}$, and:

$$\mathbf{\Phi}(t_b, t_a) = \begin{bmatrix} \mathbf{I} & (t_b - t_a)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.8}$$

is the state transition matrix from time t_a to time t_b. This factor is derived from a noise-on-acceleration model of dynamics, encouraging a zero acceleration (and therefore a feasible and smooth) trajectory.

3.4.3 Obstacle Factor

A unary factor f_o is connected to each robot state representing its distance from static obstacles in the environment [Mukadam et al., 2018b]. A pre-computed 2D signed distance field (SDF) of the environment is used. The obstacle factor is defined as:

$$\boldsymbol{h}_o\left(\mathbf{x}_k\right) = \begin{cases} 1 - \frac{d_o(\boldsymbol{x}_k)}{r_R} & d_o(\boldsymbol{x}_k) \leq r_R + \epsilon \\ 0 & \text{otherwise} \end{cases} , \tag{3.9}$$

$$\mathbf{\Lambda}_o = \sigma_o^{-2} \mathbf{I} \, , \tag{3.10}$$

where $d_o(\boldsymbol{x}_k)$ is the signed distance of point $\boldsymbol{x}_k$ from the nearest static obstacle and r_R is the robot radius. It is assumed that a robot can obtain the signed distance instantaneously from the SDF.

3.4.4 Inter-Robot Factor

If a robot A encounters another robot B within its communication range r_C, inter-robot factors f_r are created for all states $\mathbf{x}_k$ of robot A's planned trajectory and their counterparts in that of robot B for $1 \leq k < K - 1$. This ensures that the two robots will plan to not occupy the same position at the same time (collide).

This factor has a non-zero cost if the distance between the robots at a particular time step is less than a critical distance $r^* = 2r_R + \epsilon$, where r_R is the radius of a robot and ϵ is a small 'safety distance'. The inter-robot factor is defined as:

$$\boldsymbol{h}_r\left(\mathbf{x}_k^A, \mathbf{x}_k^B\right) = \begin{cases} 1 - \frac{d_r(\boldsymbol{x}_k^A, \boldsymbol{x}_k^B)}{r^*} & d_r(\boldsymbol{x}_k^A, \boldsymbol{x}_k^B) \leq r^* \\ 0 & \text{otherwise} \end{cases}, \tag{3.11}$$

where:

$$d_r(\boldsymbol{x}_k^A, \boldsymbol{x}_k^B) = \left\| \boldsymbol{x}_k^A - \boldsymbol{x}_k^B \right\| \tag{3.12}$$

is the distance between the two robots at time t_k. Here, we augment the notation for robot states with the superscripts A and B to distinguish between the two robots. The factor precision matrix takes the form $\boldsymbol{\Lambda}_r = (t_k \sigma_r)^{-2}\mathbf{I}$, such that the factor is weaker for states that are further in the future.

Figure 3.2 shows that there are two inter-robot factors for each time step of the trajectories. This symmetry in factor creation represents a shared responsibility between the two robots for planning safe trajectories but is also a redundancy in the design. In future work this could be extended further to allow heterogeneous robots to define their own trajectory parameters, such as safety thresholds or self-importance.

3.4.5 Online Algorithm

Each robot performs Algorithm 1. At each time step, the current state $\mathbf{x}_0$ of each robot is updated using the simulation time step duration Δt. In the case of a moving horizon, the horizon state $\mathbf{x}_{K-1}$ is updated similarly.

When new robots are within communication range r_C, inter-robot factors are created between them if they do not already exist, and destroyed when the robots are out of range.

3.4.5.1 Internal vs Inter-Robot GBP

Robots perform iterative trajectory optimisation using GBP under two regimes of message passing, *internal* (between variables and factor nodes in their own factor graph) and *inter-robot* (between their nodes and those of neighbouring robots). This separation reflects the different physical bottlenecks in the system: local computation versus wireless communication.

- **Internal GBP.** M_I iterations of message passing are conducted between variables and factors that are stored on a single robot, namely its trajectory states and their associated pose, dynamics and obstacle factors. These messages can be computed at a high rate using only on-board resources, without consuming any communication bandwidth. We imagine that an individual robot can be continuously performing internal message passing, refining its own short-horizon trajectory against the current estimate of all local constraints. In practice, a large M_I makes the robot's behaviour close to that of a centralised optimiser.
- **Inter-Robot GBP.** M_R iterations of message passing are conducted between the states of one robot that are connected to the states of another robot via the inter-robot factors f_r. Each iteration requires exchanging belief information across robots and then recomputing messages that enforce the shared collision-avoidance constraints. Since each inter-robot iteration consumes wireless bandwidth, incurs latency, and is vulnerable to packet loss, M_R is typically kept small compared to M_I. Conceptually, M_R controls how much a robot negotiates with its neighbours.

Intuitively, M_I controls how well each robot optimises its own planned path given the latest information from its neighbours, while M_R controls the level of coordination achieved across robots. With very few inter-robot iterations, robots can still produce smooth, dynamically feasible paths on their own, but these plans may be only loosely coupled, which can lead to conservative behaviour or local deadlocks in dense settings. Increasing M_R

allows more thorough negotiation of their planned paths: in tightly packed formations, narrow passages or high-flow junctions, extra inter-robot message passing lets robots jointly plan several steps ahead around one another, reducing travel time and creating smoother planned paths. Conversely, in sparse environments or under strict communication limits, it can be better to keep M_R small and rely mainly on extensive internal GBP, accepting weaker coordination in order to reduce communication overhead.

Algorithm 1 For one robot R_i

1: Update $\mathbf{x}_0$ and $\mathbf{x}_{K-1}$ by Δt.
2: *Let* $C(R_i)$ *be a set of robots currently connected to* R_i.
3: *Let* $N(R_i) = \{R_j \mid \|R_i - R_j\| < r_C\}$ *be a set of robots within the communication radius of* R_i.
4: **while** True **do**
5: **for** $R_j \in N(R_i) \backslash C(R_i)$ **do**
6: Create inter-robot factors $f_r(R_i, R_j)$ with a newly observed robot R_j.
7: **for** $R_j \in C(R_i) \backslash N(R_i)$ **do**
8: Delete inter-robot factors $f_r(R_i, R_j)$ with a robot R_j out of range.
9: Perform M_I internal and M_R inter-robot GBP iterations to adapt trajectory to the newly updated states and information.

3.5 Evaluation

Our method for multi-robot planning is generalised and can be applied to many scenarios in which robots traverse a 2D environment. We consider scenarios where coordination between robots is of importance.

In the first experiment robots of various sizes begin in a circle formation and travel to the antipodal sides. We also investigate the effect of obstacles and the communication range of robots. In the second experiment we investigate the performance of our algorithm for robots traversing a junction, typical of highly coordinated robots in a warehouse space. Finally we highlight the robustness of GBP planning to communications failure due to dropped messages.

3.5.1 Baseline

We compare our method with Optimal Reciprocal Collision Avoidance (ORCA) [van den Berg et al., 2009] which is a popular multi-agent distributed collision avoidance system that can be used in problems requiring a high degree of coordination. With ORCA each robot uses Linear Programming (LP) to optimise its instantaneous velocity based on the constraints imposed by the velocities of its neighbours. If no solution can be found ORCA chooses the velocity that violates the least number of constraints.

Our method is also distributed and can react to new information as the robot follows its trajectory but unlike ORCA is a short term state planner. In our experiments we consider robots moving in tight spaces at speeds an order of magnitude *higher* than those used in the ORCA demonstrations. As such there were three main parameters in the ORCA algorithm that we tuned for our experiments to allow for a fair and meaningful comparison to our method.

`neighbourDist` The maximum distance to other robots that a robot takes into account when optimizing for a new safe velocity. The larger this number the greater the number of constraints on the robot. For a fair comparison of ORCA with our method we set this parameter to be equal to the communication range r_C.

`timeHorizon` The smallest time into the future for which collision-free paths are guaranteed. We set this to the value of the horizon T_{K-1} used in our method.

`timeStep` If the simulation time step was too large, collisions occurred owing to the high speeds of the robots and so a value of $\Delta t = 0.05$ s was used for ORCA.

3.5.2 Metrics

We compare metrics for efficiency and smoothness of trajectories. Robots working together in high-coordination problems must minimise their distance travelled as well as the total time taken to reach their goals. The planned trajectories (which are constantly being replanned) must also be smooth and collision-free. As a robot moves through the environment we calculate the following metrics:

Distance travelled. The total distance travelled by the robot until its goal is reached. Efficient trajectories should minimise this metric.

Makespan. The total time taken for all robots to reach their goals. A system of many robots working together efficiently should minimise this metric.

Smoothness. Robot trajectories must be smooth, in order to be feasible with respect to real-world constraints such as those due to actuators. Since smoothness is intrinsically a geometric property of the path, it should not depend on time taken or velocity. We use the Log Dimensionless Jerk (LDJ), a smoothness metric first defined in [Balasubramanian et al., 2015]. This metric is a log-normalised value of the total squared jerk along the trajectory and is defined as

$$LDJ \triangleq -\ln\left(\frac{(t_{final}-t_{start})^3}{v_{max}^2}\int_{t_{start}}^{t_{final}}|\ddot{v}(t)|^2\,dt\right), \tag{3.13}$$

where $t \in [t_{start}, t_{final}]$ is the time interval over which the metric is considered, $v(t)$ is the robot velocity at time t, and v_{max} is the maximum velocity along the trajectory. The LDJ metric was shown to be a better indicator of smoothness than other dimensionless metrics, and values that are more positive represent 'smoother' paths.

3.6 Results and Discussion

For the following experiments we set the simulation time step to be $\Delta t = 0.1$ s for our GBP planner. Inter-robot and internal GBP was conducted with $M_R = 10$ and $M_I = 50$ iterations per time step respectively. We also set $\sigma_p = 1 \times 10^{-15}$ m and $\sigma_r = \sigma_o = 0.005$. We encourage the reader to view the accompanying video demonstration at https://aalpatya.github.io/gbpplanner .

3.6.1 Circle Experiment

Robots of various sizes are initialised in a circle formation of radius 50 m, with an initial speed of 15 m/s, towards a stationary horizon state at the opposite side of the circle. The radii of the robots are sampled randomly from $r_R \sim \mathcal{U}(2, 3)$ m. We set t_{K-1} as the ideal time taken for a single robot moving from the initial speed to rest across the circle at constant acceleration. This would correspond to a smooth (zero jerk) trajectory. Each robot has a communication range $r_C = 50$ m representing a partially connected network of robots. In addition, we set $\sigma_d = 1$ m.

This is a complex problem requiring coordination among many robots. Figure 3.1 shows the paths taken by GBP planning and ORCA when $N_R = 10$. GBP planning allows robots to collaborate and reach their goals at similar times.

As the number of robots N_R is varied, figure 3.4 shows the distribution of robot distances travelled. With the GBP planner, path lengths were close to 100 m (the diameter of the circle) and were shorter than with ORCA. The spread of robot distances travelled was also smaller due to inter-robot collaboration, whereas in ORCA each robot acted for itself, choosing a path at the expense of longer paths for others.

Figure 3.5 shows the distributions of the LDJ metric at each N_R. GBP planning creates

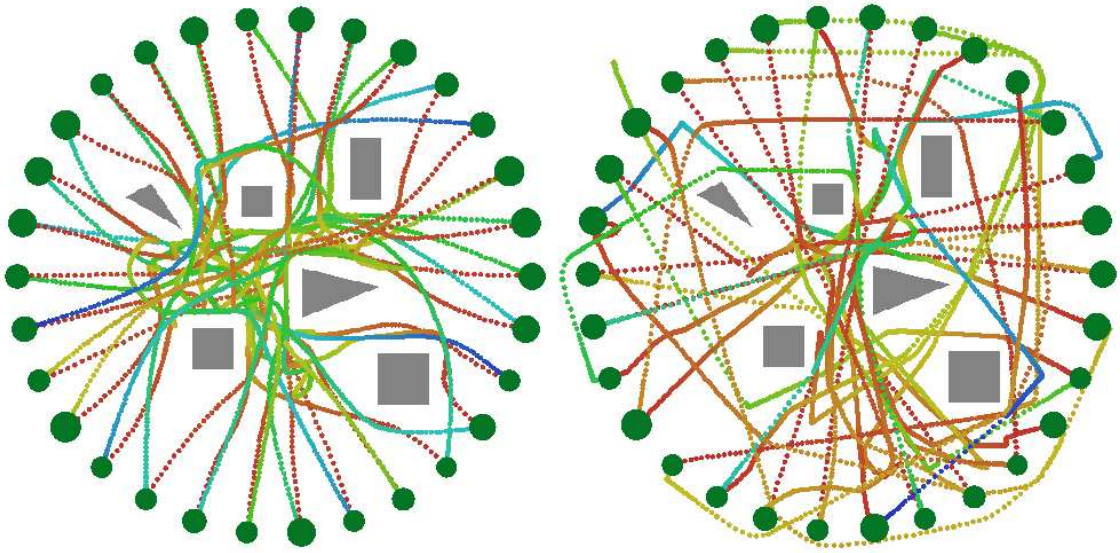


Figure 3.3: Circle experiment with obstacles for $N_R = 30$ robots for the GBP planner (left) and ORCA (right). Colours change along the paths with time, from red (oldest) to blue (newest).

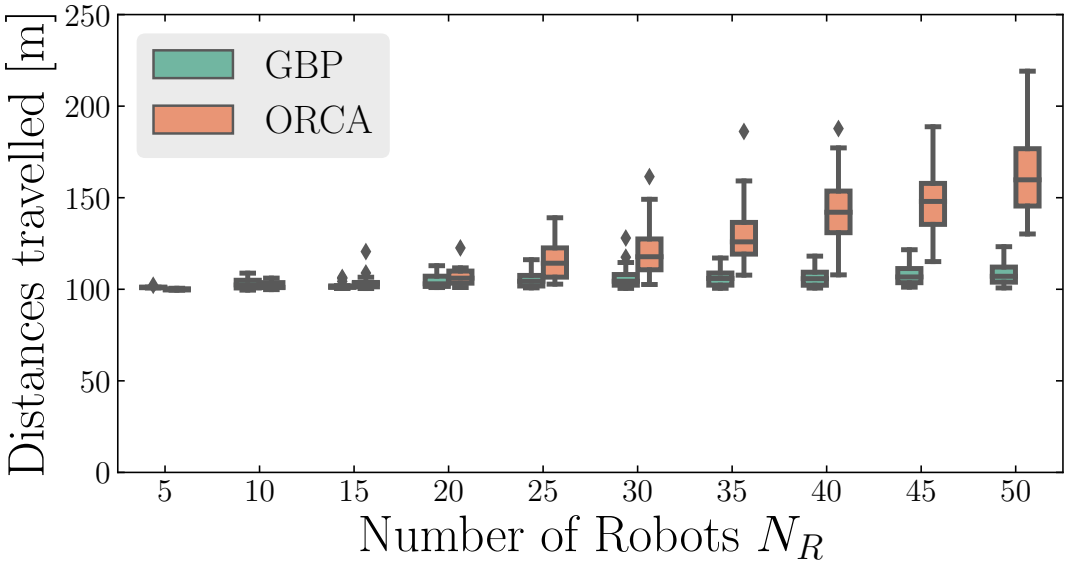


Figure 3.4: Circle experiment: distribution of distances travelled as number of robots in the formation N_R is varied. The GBP planner creates shorter paths and a smaller spread of distances than ORCA; robots collaborate to achieve their goals.

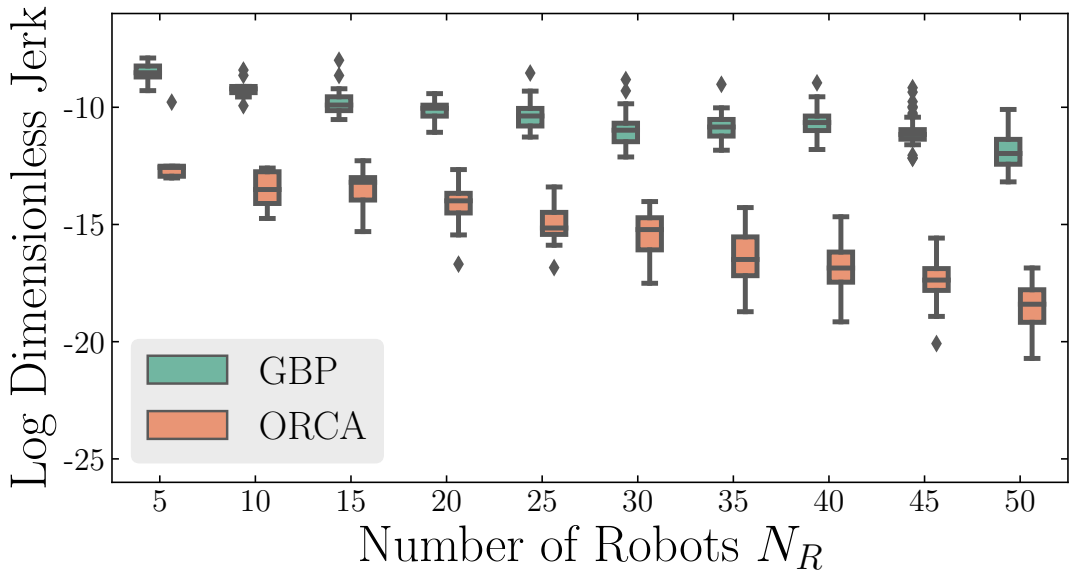


Figure 3.5: Circle experiment: distribution of the LDJ metric as N_R increases, with smoother trajectories shown by more positive values. The worst performing GBP planning robots had smoother paths than the best robots for ORCA.

trajectories that are far smoother than ORCA. As N_R increases, this difference in performance between our method and ORCA increases. In general the *worst performing* GBP planning robots had trajectories that were smoother than the *best performing* robot using ORCA.

As we are presenting a state planner we assume that robots can perfectly execute their planned trajectories. With real-world robots this can be helped if the planned trajectories are smooth, efficient and induce minimal jerk.

The solid lines in figure 3.6 depict the makespan as N_R is varied. This metric increases in an approximately linear fashion with GBP, and performs better than ORCA for higher N_R. ORCA performs better at when N_R is low — robots reach their goals in shorter amounts of time. However this is at the expense of unrealistic jerky trajectories.

We consider two further experiments in the circle scenario.

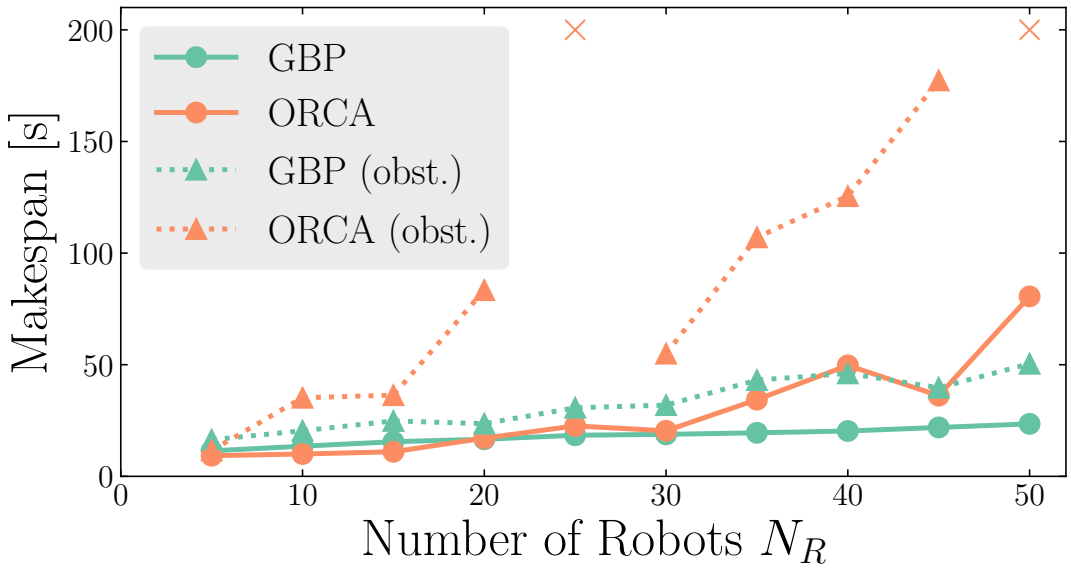


Figure 3.6: Circle experiment with (solid) and without (dotted) obstacles: makespan as N_R increases. Crosses denote trials where not all robots reached their goals.

3.6.1.1 Environment obstacles

Also shown in figure 3.6 with dotted lines are the makespans for ORCA and our GBP planner when 5 polygonal obstacles are placed in the middle of the circle. The paths can be seen in figure 3.3. The results are for one layout of obstacles averaged over 5 seeds. For $N_R = 25$ and 30 some robots using ORCA became deadlocked with the obstacle configuration. Our method performs well with obstacles, producing makespans that are only slightly higher than in those in free space.

Table 3.1: Effect of varying communication range r_C in the circle experiment with obstacles

r_C [m]	Makespan [s]	Mean distance travelled [m]	Log Dimensionless Jerk
20	12.0	104.0	-9.02
40	12.3	104.5	-8.76
60	12.7	104.0	-8.38
80	14.8	103.8	-8.47

3.6.1.2 Varying Network Connectivity

Robots within a communication range r_C of each other form a partially connected network, and can collaboratively modify their planned paths. We investigate the effect of varying r_C for $N_R = 30$ for the 100 m diameter circle formation with obstacles. Table 3.1 shows that as r_C increases robots take more of their neighbours into account, resulting in greater makespans but small changes in the distances travelled and path smoothness. This highlights the applicability of our method to real networks where sensing and communication range may be limited.

3.6.2 Junction Experiment

Robots working in crowded environments may have to operate at high speeds with high levels of coordination e.g. traversing junctions between shelves a warehouse. We simulate one such junction with channel widths of 16 m and robots moving at 15 m/s with horizon $t_{K-1} = 2$ s. We set $\sigma_d = 0.5$ m.

Multirobot systems must be able to maintain a high flowrate of robots without blockages occurring in the junction. We vary the rate Q_{in} at which robots enter the central section of the junction and measure the rate Q_{out} at which they exit and compare our method against ORCA. We choose the central section of the junction to measure flow over 500 time steps to represent steady-state flow behaviour. Robots must exit the junction in the same direction that they entered, without collision; we refer to this as the 'correctness condition'.

Figure 3.7 shows Q_{out} against Q_{in} for GBP planning and ORCA. A dashed line representing the ideal case of equal input and output flowrates ($Q_{out} = Q_{in}$) is also shown for comparison. Trials where the correctness condition was violated are shown as crosses. GBP planning is able to sustain flowrates of up to $Q_{in} = 5.5$ robots/s (figure 3.8), com-

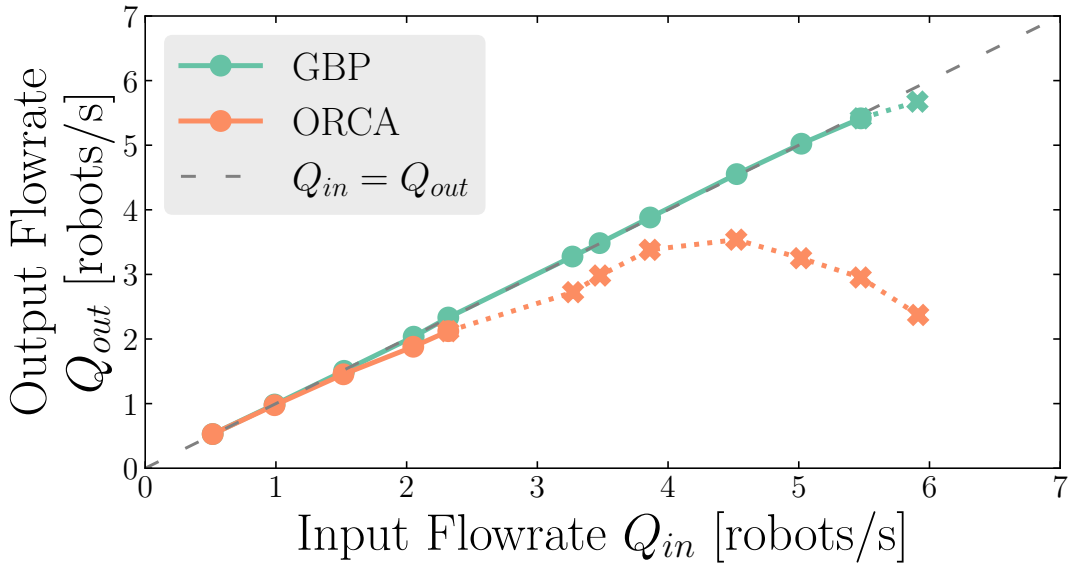


Figure 3.7: Junction experiment: input vs output flowrates. Robots cross the junction at 15 m/s. Points shown as crosses denoted trials where the correctness condition was violated. GBP planning can sustain flowrates of robots up to $Q_{in} = 5.5$ robots/s, compared to ORCA which breaks down at $Q_{in} = 2.3$ robots/s.

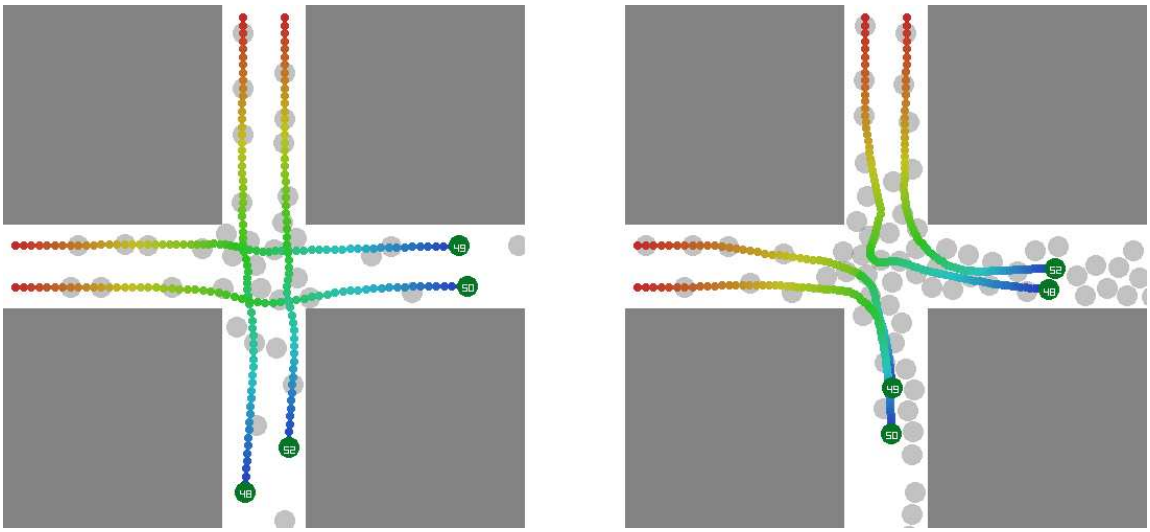


Figure 3.8: Qualitative comparison of our GBP planner (left) and ORCA (right) in the junction experiment at a high ($Q_{in} = 5.5$ robots/s) flowrate. Selected robots are shown in colour. With GBP planning robots can smoothly swerve past each other whereas with ORCA robots block the junction and are deflected.

pared to ORCA which begins to break down at $Q_{in} = 2.3$ robots/s. This is due to GBP planning allowing robots to communicate, and to collaboratively negotiate their planned trajectories with neighbouring robots before they are close to each other.

It is important to note that at high enough flowrates, the effectiveness of both GBP planning and ORCA breaks down due to the dense packing of incoming robots.

3.6.3 Communications Failure Experiment

Our GBP planner relies on per-time-step peer-to-peer communication between robots. It is assumed that each robot follows a protocol similar to [Murai et al., 2024b]; it always broadcasts its state information. We consider a communications failure scenario where a robot is not able to receive messages from robots it is connected to. We would expect more cautious behaviour when planning a trajectory.

We simulate a communication failure fraction γ: at each time step the robot cannot receive any messages from a randomly sampled proportion γ of its connected neighbours. We repeat the circle experiment with 21 robots at two different initial speeds of 10 m/s and 15 m/s, measuring the makespan. The reported result is an average over 5 different random seeds. To be fair, at any time step for any robot, the failed communications are exactly the same given a fixed seed for both initial velocities considered.

Table 3.2 shows that as γ increases, it takes longer for all robots to reach their goals. However, trajectories for the 10 m/s case are completely collision free up to $\gamma = 80\%$. As the initial speed increases, collisions happen at lower values of γ as robots have less time to react to faster moving neighbours who they may not be receiving messages from.

This experiment shows one of the benefits of GBP — safe trajectories can still be planned even with possible communication failures, which is likely in any realistic settings.

Table 3.2: A comparison of makespans for the circle experiment with 21 robots for two initial velocities under varying rates of communication failure. Values for mean number of collisions were taken over 5 random seeds.

Initial speed [m/s]	10		15	
Comm. failure γ [%]	Makespan [s]	Mean num. collisions	Makespan [s]	Mean num. collisions
0	19.5	0.0	14.9	0.0
10	20.3	0.0	17.1	0.0
20	22.9	0.0	18.9	0.0
30	25.7	0.0	22.5	0.0
40	30.8	0.0	26.5	0.0
50	35.6	0.0	30.6	0.0
60	42.0	0.0	38.8	0.2
70	51.3	0.0	44.6	0.8
80	87.4	0.0	63.4	0.8
90	146.9	1.6	12.6	4.6

3.7 Conclusions

We have shown that our collaborative method for short-term trajectory planning using Gaussian Belief Propagation (GBP) can ensure smooth, efficient and safe trajectories in problems where a high degree of coordination is required. By performing message passing between robots in a purely peer-to-peer manner, there is no need for a centralised solver; the distributed nature of the problem could be scaled up to large numbers of robots.

Our current approach, like most other planners, assumes that all robots have perfect knowledge of their localisation. However it is clear that our planning approach could be combined with the multi-robot localisation method based on GBP in [Murai et al., 2024b] which uses the same communication and computation pattern, to enable true infrastructure-free multi-robot operation.

In the future, we will explore other applications of GBP trajectory planning, such as the coordination of larger numbers of robots which can organise themselves into useful formations.

Chapter 4

GBPStack: A Distributed Multi-Robot Framework for Exploration, Information Acquisition and Consensus

Meditate often on the interconnectedness and mutual interdependence of all things in the universe ... all things are mutually woven together.

— Marcus Aurelius, Meditations 6.38

Contents

While Chapter 3 focused on a path planning task, real multi-robot systems must often balance several interdependent objectives. This motivates the extension of GBP to problems spanning multiple competencies, each represented in a structured hierarchy of factor graphs. In this chapter we present a framework for structured optimisation, enabling teams of robots to coordinate across layered sub-problems in a principled manner.

4.1 Introduction

Multi-robot teams offer the potential of solving complex tasks beyond the abilities of single robots, and for scalability it is desirable to seek distributed solutions where the robots operate with local computation and peer-to-peer communication rather than centralised control. The fact that robot teams are not yet widely used in real applications perhaps reflects the many aspects of competence that are needed to achieve whole tasks, and the difficulty in providing distributed, yet connected solutions to these. For instance a robot team tackling an environmental inspection task must be capable of localisation, planning,

mapping and coordinated active information acquisition. Each one of these competences on its own is difficult to achieve in a distributed manner, but for the whole task to be performed well they must also influence each other: localisation is needed for planning, which in turn is needed for coordinated information acquisition.

In this work we show that various types of competence for robot teams can be modelled as *factor graphs*, and that this formulation allows for straightforward and principled coupling between competencies as layers in a graph stack.

The novelty of our method is to show that the various competencies or problem domains that influence each other can be optimised independently and asynchronously. Robots effectively hold and update a local copy of a global state based on ad-hoc collaboration with other robots, reaching consensus on its true value.

We build on recent work which showed that Gaussian Belief Propagation (GBP) is a powerful general tool for distributed factor graph inference, specifically when applied to multi-robot planning [Patwardhan et al., 2023].

Although our formulation for such a joint-optimisation problem is general, we demonstrate it here in simulations of multi-robot exploration and information acquisition about a global state. In the first scenario a swarm of robots must search for a target region in the environment. In the second, the robots must collectively explore the whole environment and accurately estimate the global state. Finally we investigate the performance of our algorithm under communication limitations.

In summary, our main contributions are:

- The first application of GBP to a multi-robot, joint optimisation problem over multiple competencies where robots must reach a consensus on a global state through local measurements.
- Qualitative and quantitative simulations to show that our method outperforms

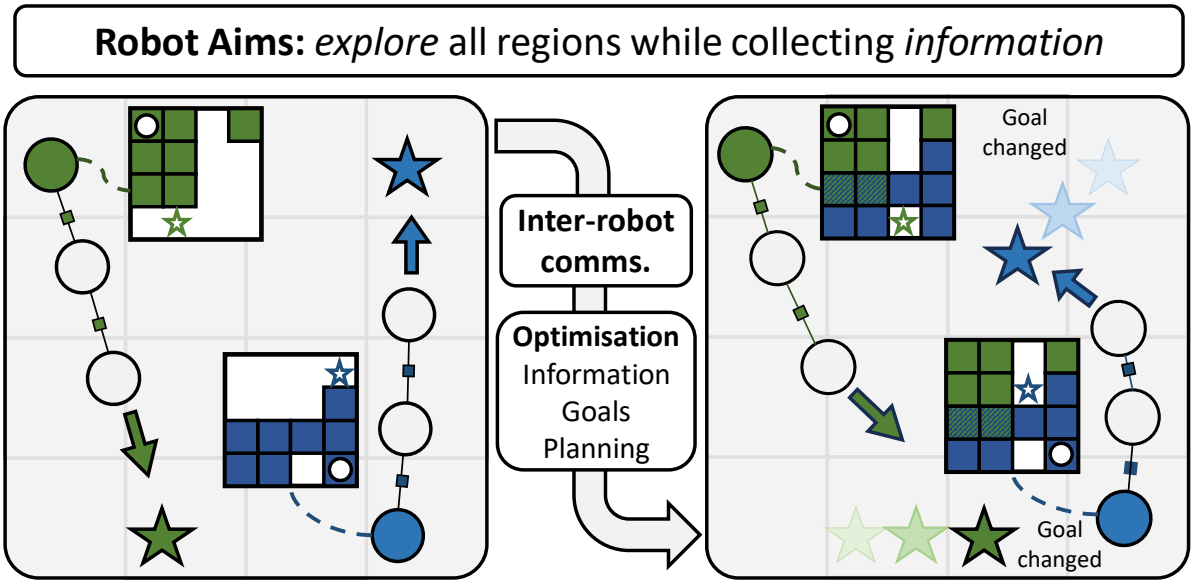


Figure 4.1: A simplified schematic of the general applications we consider. Left: two robots (circles) have planned their paths towards goals (stars) in unexplored regions based on partial information they have estimated about the global state (shown here as a grid for each robot representing exploration. Right: after message passing they have combined and improved their estimates, optimising their planned paths towards new regions.

related work in swarm source-seeking, and the scalability and robustness of our algorithm under communication limitations.

4.2 Related Work

We are interested in planning for multi-robot teams which have coordinated goals, such as efficient, collaborative information acquisition. Information-based planning is a challenging problem with wide-ranging applications such as gas distribution mapping [Atanasov et al., 2014], environmental monitoring [Woosley et al., 2020], or target search. Non-myopic information acquisition methods for single robots can be categorised into search-based and sampling-based methods. Search-based methods can find globally optimal paths by searching the motion and information spaces at the expense of computational

efficiency [Le Ny and Pappas, 2009]. Sampling-based methods however are able to find solutions, but relax guarantees about optimality by considering subsets of the search space. The centralised algorithm in [Kantaros et al., 2021] explores the motion and information spaces by incrementally creating a directed tree, biasing sampling towards informative regions of the environment.

Distributed solutions for multiple robots offer the clear potential for much faster and more scalable information acquisition, but require coordination and communication. The semi-distributed algorithm for gas-plume tracking in [Kapoutsis et al., 2021] adjusts a swarm formation to maximise the combined perception of the swarm around areas of high gas concentrations. A centralised unit aggregates measurements made by robots and assigns tasks to each of them to maximise the combined perception of the plume around areas of high gas concentration. If global communication between robots can be assumed, algorithms based on Particle Swarm Optimisation (PSO) can be used which model robots as particles performing random walks biased towards the collective best observation regions. These methods do not guarantee success but have been shown to operate well in real-world experiments [Duisterhof et al., 2021] where a swarm of small drones localise a gas source in an unknown cluttered environment by communicating observed gas concentrations with the group. Without the guarantee of global connectivity, efficient information dissemination techniques are needed once a robot has completed its task. The hybrid PSO method in [Ebert et al., 2022] simulated robots exploring a complex environment in the form of Perlin noise, searching for a target region below a given threshold. Once a suitable region was found, robots aimed to maximise dissemination of the information across the group with minimal coordination.

These approaches do not consider robot dynamics or noisy signal measurements in their optimisation and produce unrealistic, jerky paths. When there are differences in the measurements by two robots of a global property due to sensor noise, they should be able to reach a consensus on the true value [Shirsat et al., 2020].

Multi-robot planning is a well studied topic in literature where robots must plan paths through an environment while avoiding each other and obstacles which may be dynamic. Local distributed planners optimise for the planned paths of robots in a short forward time window, and allow for efficient re-planning of paths in reaction to changes in the environment through communication with other robots [Zhou et al., 2021], [Luis et al., 2020].

Recently, Gaussian Belief Propagation (GBP) has been shown to be an ideal tool for distribution of convergent computation between multiple robots, whether for multi-robot localisation [Murai et al., 2024b] or short-horizon avoidance planning [Patwardhan et al., 2023] and can offer efficient solutions. However, in that work, the multiple robots all had *separate*, selfish goals. In this chapter, for the first time we show that an integrated, multi-layer factor graph approach with GBP can be used to achieve coordinated multi-robot behaviour for information acquisition. The different elements of the whole system — local avoidance planning, global information-guided goal coordination and coordinated, convergent map creation, can all be formulated as linked factor graphs.

4.3 Technical Background

4.3.1 Gaussian Belief Propagation (GBP)

Once again we turn to GBP to perform inference on factor graphs as introduced in 2.3. In this chapter we use a multi-layered factor graph to represent an abstraction for multi-robot planning problems that are dependent on information acquisition about a global state as illustrated in Figure 4.3. We will explain the details of the factors shown in the sections that follow.

4.4 Method

We consider the complex problem of exploration and information acquisition where a swarm of robots $\mathcal{R} = [\mathrm{R}_i],\ i \in [0, N_R - 1]$ must efficiently explore an environment $\mathcal{M} \in \mathbb{R}^d$ to collaboratively estimate a global state $\mathbf{\Psi}$.

In general $\mathbf{\Psi}$ can represent any scalar or vector quantity present throughout the environment with values in the range [0,1] of which robots can make noisy and partial measurements. In our work $d = 2$ and $\mathbf{\Psi}$ represents a scalar signal field such as gas concentration present throughout the environment $\mathcal{M}$ which is segmented into N_M square sampling regions of width r_Dm. Robots can make measurements of $\mathbf{\Psi}$ for regions $m \in \mathcal{M}(r_S)$ within their sampling radius r_S.

The aims of the robots are to:

- Collaborate with other robots to form a consensus on the global state using noisy local measurements.
- Find a location where the signal value is below a given threshold, or collectively explore the whole environment.
- Avoid collisions with other robots and plan smooth paths over a time horizon T_K in Euclidean space.

A simplified schematic of the problem is shown in Figure 4.1.

4.4.1 A GBP Stack of Factor Graphs

Robots plan paths towards intermediate goal locations in the environment which directly influence the information they collect about $\mathbf{\Psi}$ through sampling. However, this collected

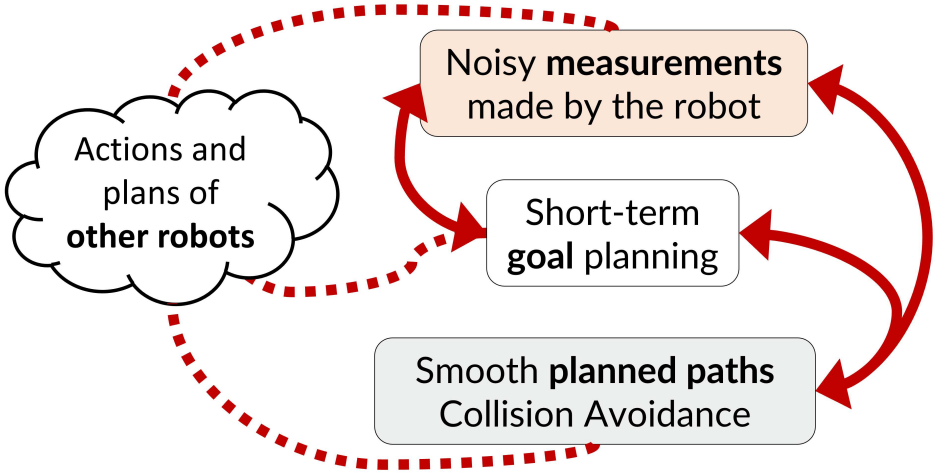


Figure 4.2: Complex tasks such as multi-robot exploration and information acquisition consist of multiple inter-dependent sub-problems or competencies; optimisation over one of these sub-problems influences the others.

information can also actively guide the path planning goals to new and potentially informative regions; there is a clear inter-dependency between information acquisition, path planning and goal selection.

We formulate the problem as a *stack of factor graphs* which we call the 'GBP stack', where the layers of the stack represent different problem domains that can be solved using GBP. Each robot holds variables and factors in the layers of its own GBP stack, and is able to create and destroy factors between themselves and their neighbours in an ad-hoc manner. In this way optimisation can be done using GBP both *within* a robot's stack as well as *between* a robot and its neighbours.

This is a general formulation, but in this work we consider a robot's GBP stack formed of *layers* representing information acquisition, goal selection, and local path planning as shown in Figure 4.3. The information layer $\mathcal{I}$ contains a robot's copy of the global state $\mathbf{\Psi}$, updated with local measurements and information from other robots. The goal layer $\mathcal{G}$ is responsible for the selection of regions to plan paths towards. Finally the planning layer $\mathcal{P}$ is responsible for the short range path planning and collision avoidance of the robot.

In real-world systems robots may be restricted in their inter-robot communication due to bandwidth or power constraints and may not receive information at every time step for each layer. The asynchronous nature of GBP allows a robot to continue optimising over $\mathcal{G}$ and $\mathcal{P}$ using the most recent information it holds in $\mathcal{I}$.

For the rest of this thesis we will use the following notation to denote the m'th GBP variable $\mathbf{x}$ belonging to GBP stack layer $\mathcal{L}$ of robot R:

$$[\text{layer } \mathcal{L}]\mathbf{x}\begin{matrix}[\text{robot } R]\\ [\text{index } m]\end{matrix}$$

If there is only one variable in a particular layer of a robot's GBP stack, m is assumed to be 0 and is dropped from the notation. Similarly, the robot index R may also be dropped for brevity if it is not necessary. We now describe in each layer in detail, and the types of variables and factors that are involved.

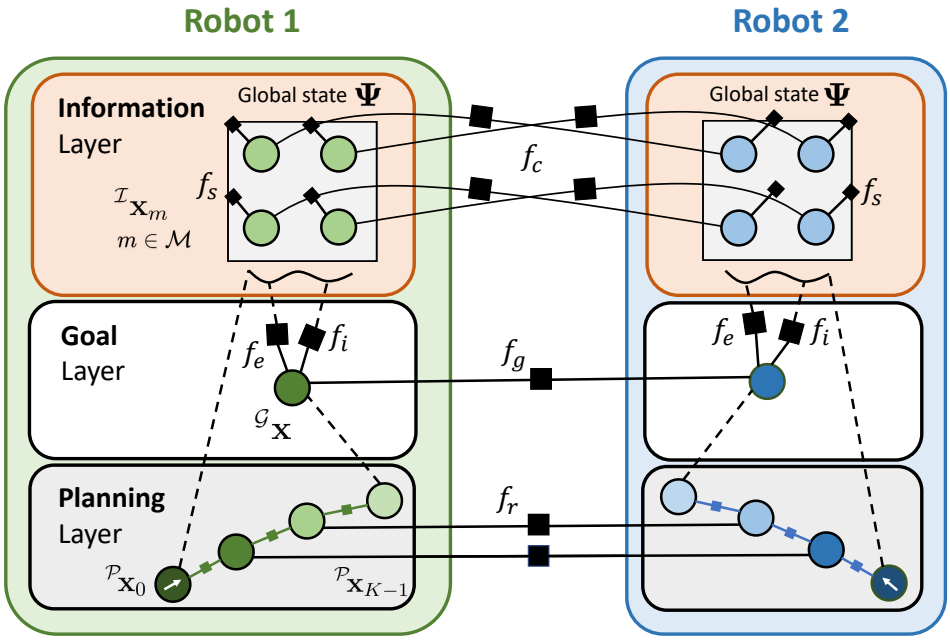


Figure 4.3: The GBP stacks (of factor graphs) for two robots estimating a global state $\mathbf{\Psi}$ shown here as being segmented into $N_M = 4$ regions. A robot holds a copy of $\mathbf{\Psi}$ in its Information layer $\mathcal{I}$ which it can update asynchronously. The Goal layer $\mathcal{G}$ guides the Planning layer $\mathcal{P}$ for exploration. Message passing can happen asynchronously between the layers of a robot's stack as well the stacks of other robots.

4.4.2 Information Layer $\mathcal{I}$

This layer represents a robot's copy of the global state; for each region m in the environment, there is one variable ${}^{\mathcal{I}}\mathbf{x}_m = [{}^{\mathcal{I}}\boldsymbol{p}_m, {}^{\mathcal{I}}\psi_m, {}^{\mathcal{I}}\zeta_m]^\top$. Dropping the superscript $\mathcal{I}$ for brevity, this variable is formed of the position $\boldsymbol{p}_m$, signal value ψ_m, and coverage $\zeta_m \in [0, 1]$ of the region m. $\zeta_m = 1$ signifies that the cell has been visited by the robot, and we set $\psi_m = 1$ at initialisation.

4.4.2.1 Sensor Factor

Each variable is connected to a unary factor representing a noisy sensor measurement $\mathbf{z}_s$ made by the robot of the region m. The factor is represented as

$$f_s : \ \boldsymbol{h}_s({}^{\mathcal{I}}\mathbf{x}_m) = {}^{\mathcal{I}}\mathbf{x}_m, \ \boldsymbol{\Lambda}_s = diag(\sigma_p^{-2}, \sigma_\psi^{-2}, \sigma_\zeta^{-2}) \ , \tag{4.1}$$

$$\boldsymbol{z}_s \sim \mathcal{N}([\tilde{\boldsymbol{p}}_m, \tilde{\psi}_m, 1]^\top \ , \boldsymbol{\Lambda}_s^{-1}) \ . \tag{4.2}$$

where the tilde $\sim$ above a variable represents its true value. In our work $\sigma_p = \sigma_\zeta = 10^{-5}$, representing perfect measurement of the location of m.

4.4.2.2 Inter-Robot Consensus Factor

Each variable in this layer is also connected to the corresponding variables in the Information layers of other robots. These factors have precision $\boldsymbol{\Lambda}_c = \sigma_c^{-2}\mathbf{I}$ and are represented as

$$f_c : \ \boldsymbol{h}_c({}^{\mathcal{I}}\mathbf{x}_m^1, {}^{\mathcal{I}}\mathbf{x}_m^2) = {}^{\mathcal{I}}\mathbf{x}_m^1 - {}^{\mathcal{I}}\mathbf{x}_m^2 \ . \tag{4.3}$$

For a pair of robots (R_1, R_2), these factors encourage R_1's beliefs of the variables to be the same as those of R_2, allowing the robots to reach a consensus on the global state $\boldsymbol{\Psi}$ through iterations of GBP, as well as exchanging information about the regions of the environment they have collectively visited.

4.4.3 Goal Layer $\mathcal{G}$

The Goal layer in a robot's GBP stack is responsible for deciding on regions to plan paths towards whilst avoiding regions that other robots are heading to. Each robot holds a variable in this layer ${}^{\mathcal{G}}\mathbf{x}$ representing a goal location for the robot's path planning layer. This variable is connected to factors representing information acquisition, exploration and inter-robot coordination.

4.4.3.1 Signal Factor

A factor f_i aims to draw the goal variable to the region m^* with the best (lowest) value ψ_{m^*} of $\mathbf{\Psi}$. It is therefore dependent on the most up-to-date knowledge from the Information layer. The factor precision matrix is $\mathbf{\Lambda}_i = \sigma_i^{-2}\mathbf{I}$ and the factor is denoted as

$$f_i : \ \boldsymbol{h}_i({}^{\mathcal{G}}\mathbf{x}; {}^{\mathcal{I}}\boldsymbol{p}_{m^*}) = \ u(m^*)({}^{\mathcal{G}}\mathbf{x} - {}^{\mathcal{I}}\boldsymbol{p}_{m^*}) \ , \tag{4.4}$$

$$u(m^*) = \ 1 - \psi_{m^*} \ , \tag{4.5}$$

$$m^* = \ \underset{m \in \mathcal{M}}{\operatorname{argmin}} \ \psi_m \ . \tag{4.6}$$

4.4.3.2 Exploration Factor

This factor f_e aims to draw the goal variable ${}^{\mathcal{G}}\mathbf{x}$ towards the nearest unexplored region and is therefore also dependent the Information layer as well the Planning layer with the robot's current position ${}^{\mathcal{P}}\boldsymbol{p}_0$. The factor precision is $\mathbf{\Lambda}_e = \sigma_e^{-2}\mathbf{I}$ and has the same form as (4.4) with

$$u(m^*) = 1, \quad m^* = \ \underset{m \in \{\underset{m' \in \mathcal{M}}{\operatorname{argmin}} \ \zeta_{m'}\}}{\operatorname{argmin}} \ || \, {}^{\mathcal{I}}\boldsymbol{p}_{m^*} - {}^{\mathcal{P}}\boldsymbol{p}_0 || \ . \tag{4.7}$$

If there are no unexplored regions, a random location is selected so that robots continue to explore.

4.4.3.3 Goal Diversity Factor

Finally ${}^{\mathcal{G}}\mathbf{x}$ is connected to the goal variables of neighbouring robots. This discourages two connected robots from moving towards the same region and therefore promotes exploratory behaviour in the swarm. The factor has precision $\mathbf{\Lambda}_g = \sigma_g^{-2}\mathbf{I}$ and is formed as

$$f_g : \ \boldsymbol{h}_g({}^{\mathcal{G}}\mathbf{x}^1, {}^{\mathcal{G}}\mathbf{x}^2) = \begin{cases} 1 - \frac{\|{}^{\mathcal{G}}\mathbf{x}^1 - {}^{\mathcal{G}}\mathbf{x}^2\|}{r_D} & \|{}^{\mathcal{G}}\mathbf{x}^1 - {}^{\mathcal{G}}\mathbf{x}^2\| \leq r_D \\ 0 & \text{otherwise} \end{cases} . \tag{4.8}$$

4.4.4 Planning Layer $\mathcal{P}$

This layer is based on our previous work [Patwardhan et al., 2023] where a robot's path consisted of variables representing its position and velocity in a short forward time window T_K. These variables were connected with factors representing dynamics f_d and collision avoidance f_r, and the planning horizon state had a fixed non-optimisable velocity.

In this work however the horizon state ${}^{\mathcal{P}}\mathbf{x}_{K-1}$ is dependent on the Goal layer, having a velocity of magnitude V_{max} towards ${}^{\mathcal{G}}\mathbf{x}$. A robot's planned path can therefore dynamically change based on the latest information of the world.

4.4.5 Algorithm

Robots initialise their GBP stacks with the variables and factors in each layer as described in sections 4.4.2 to 4.4.4. At every time step, each robot follows Algorithm 2.

When a new robot R_j is within communication range r_C of robot R_i, inter-robot factors are created between the two robots for each layer in the robots' GBP stacks if they do not already exist, and are destroyed when the robots are out of range.

Algorithm 2 For each robot R_i

Initialise GBP stack layers $\mathcal{I}, \mathcal{G}, \mathcal{P}$
while $t < T_{max}$ **do**
  Let $N(R_i) = \{R_j : ||{}^{\mathcal{P}}\boldsymbol{p}_0^i - {}^{\mathcal{P}}\boldsymbol{p}_0^j|| < r_C\}$ *be the set of robots within the communication radius of* R_i.
  Let $C(R_i)$ *be the set of robots connected to* R_i.
  for Newly observed robot $R_j \in N(R_i) \backslash C(R_i)$ **do**
    Create inter-robot factors f_c, f_g, f_r.
  for Out-of-range robot $R_j \in C(R_i) \backslash N(R_i)$ **do**
    Delete inter-robot factors f_c, f_g, f_r.
  Make local measurement of ${}^{\mathcal{I}}\mathbf{x}_m^i \ \forall \ m \in \mathcal{M}(r_S)$
  Activate variables ${}^{\mathcal{I}}\mathbf{x}_m^i \ \forall \ m \in \mathcal{M}(r_C)$.
  for layer $\mathcal{L}$ in GBP Stack **do**
    Perform N_I iterations of GBP
  Update ${}^{\mathcal{P}}\mathbf{x}_0^i$ and ${}^{\mathcal{P}}\mathbf{x}_{K-1}^i$ by Δt.

The robot samples the environment for regions $m \in \mathcal{M}(r_S)$ and updates its corresponding Information layer variables ${}^{\mathcal{I}}\mathbf{x}_m^i$. Joint optimisation is performed over all layers using GBP for N_I iterations. In the Information layer, only variables representing regions within the robot's communication radius r_C are set to 'active' and take part in inter-robot message passing.

Robots perform inter-robot message passing at time intervals of $T_C^{\mathcal{P}}, T_C^{\mathcal{G}}$ and $T_C^{\mathcal{I}}$ seconds for the Planning, Goal, and Information layers respectively. Finally the robot's position and horizon states in its planning layer are updated using the simulation time step Δt.

4.5 Experiments

We model a swarm of N_R robots with radius $r_R = 1$m and planning horizon $T_K = 1$s traversing a square environment of length D m, segmented into cells with side length $r_D = 10$m. In the experiments that follow, the robots have a maximum speed of $V_{max} = 5$m/s, and make measurements every 1s within a sampling radius $r_S = 10$m. The communication intervals for the Goal and Information layers of the robots' GBP stacks are set to $T_C^{\mathcal{I}} =$

$T_C^{\mathcal{G}} = 1$s, and $T_C^{\mathcal{P}} = \Delta T = 0.1$s. We set $N_I = 5$, $\sigma_c = 1$ [-], $\sigma_e = 0.1$ m, $\sigma_i = 0.5$m and $\sigma_g = 0.01$ [-], encouraging exploration whilst seeking regions of low ψ.

4.5.1 Baseline for Comparison

We compare our method with the Hybrid Particle Swarm Optimisation (PSO) algorithm of [Ebert et al., 2022], where a swarm of robots explores a complex environment with a scalar signal field searching for a region with signal value less than a given threshold. Similar to our work, the authors consider a dynamic graph topology of robot connectivity and the method operates in a purely decentralised manner. However it does not account for robot collision avoidance and creates unrealistic jerky paths. For a fair comparison we use this work as a baseline in the source seeking experiment to show the validity of our method against the same complexities of 2D Perlin noise environments and initial robot configurations. We use the best performing parameters from the work: $c_{p,g} = 1, \omega = 0$.

4.5.2 Source Seeking

Robots begin in one corner of the environment with $D = 100$m and search for a region where the value of the global state ψ is less than a threshold $\psi^* = \frac{10}{255}$ (i.e. a 'source'). The noise on their measurements of ψ is set to $\sigma_\psi = 0.01$ for our method while Hybrid PSO assumes perfect sensing.

We measure the time taken for all robots to have found or gained knowledge of the target region as we vary the number of robots N_R and the radius of communication r_C.

Figure 4.4 shows that as N_R and r_C increase the completion time decreases as robots quickly spread and cover a larger area of the environment, due to their goal diversity factors f_g as well as being able to share information about distant regions of the environment. A larger N_R results in diminishing returns for completion times due to the increase

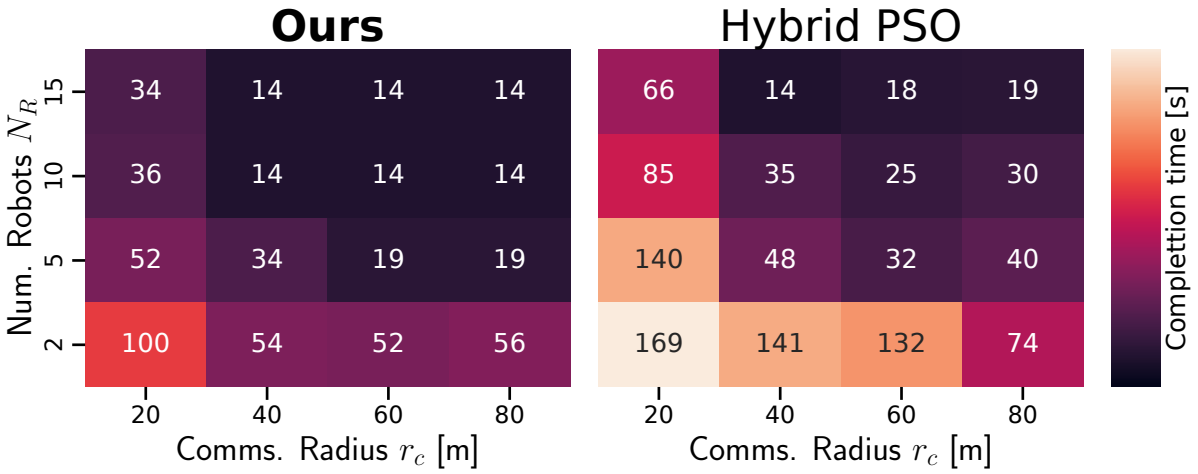


Figure 4.4: Completion times for the source seeking experiment as the number of robots N_R and communication radius r_C varies. Our method is faster than Hybrid PSO while being able to consider noisy sensor measurements.

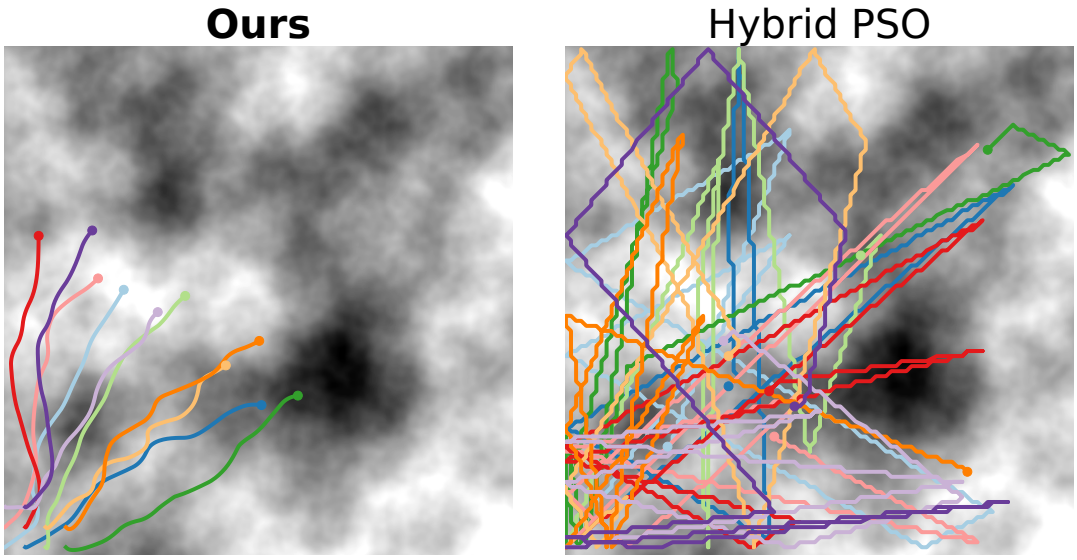


Figure 4.5: Paths taken until the target region was found in the source seeking experiment for $N_R = 10$ robots and communication radius $r_C = 40$ m. Our method optimises for smooth paths and efficient coverage of the environment without collisions.

in robot density in space. Our method outperforms the baseline due to its efficient information sharing and path planning and is able to create smooth paths as seen in Figure 4.5.

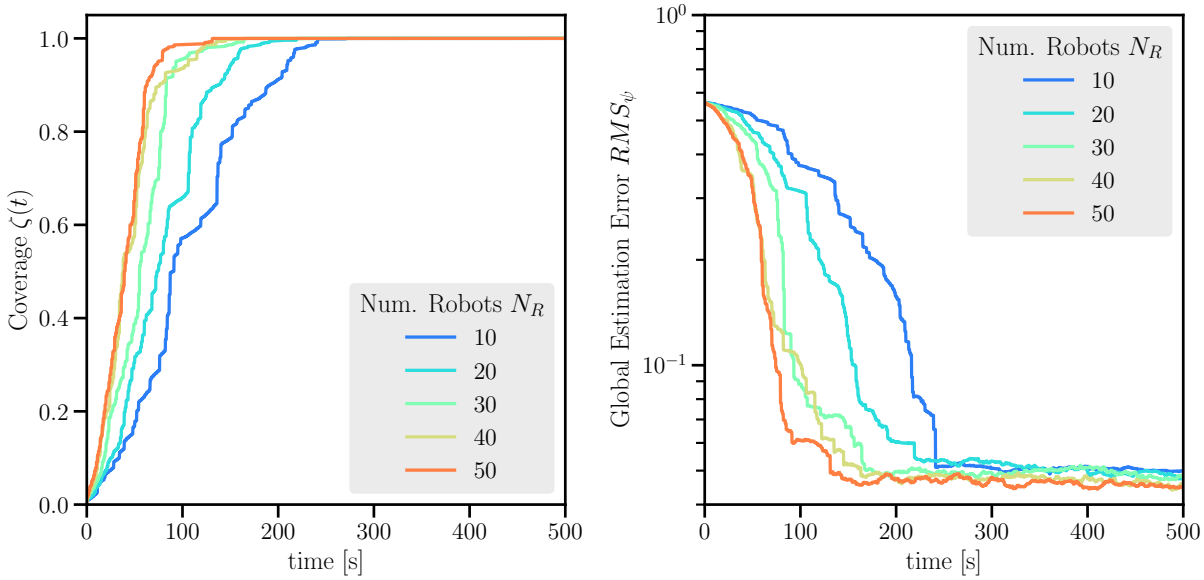


Figure 4.6: Evolution of coverage and RMS_ψ over time for the 'random' initialisation scenario. Increasing the number of robots N_R results in quicker coverage and a steady-state RMS_ψ much lower than the measurement noise ($\sigma_s = 0.1$).

4.5.3 Exploration and Coverage

In this experiment, the task of the swarm is to collectively explore an environment of width $D = 200$ m, taking noisy local measurements of the global signal field $\boldsymbol{\Psi}$ with $\sigma_\psi = 0.1$. We weaken the signal factor $\sigma_i = 1000$ whilst keeping the other factors of the same strength to encourage exploration. We vary the number of robots N_R in the swarm and consider the evolution of two metrics over time:

Coverage $\boldsymbol{\zeta}(t)$. The proportion of the environment visited ($\zeta_m > 0$) by a robot, averaged over all robots.

Global Estimation Error $\boldsymbol{RMS}_\psi(t)$. The root-mean-square difference between a robot's beliefs of ψ_m and the true values $\tilde{\psi}_m$ averaged over all regions $m \in \mathcal{M}$ and all robots.

Table 4.1: Effect of varying communication range r_C for $N_R = 20$ in the 'corner' and 'random' initialisations

	Coverage Time [s]		Steady-state RMS_ψ	
r_C	Corner	Random	Corner	Random
20	147	232	0.0408	0.0425
50	84	56	0.0375	0.0381
100	78	48	0.0359	0.0343

Robots are initialised in random positions with zero velocity, and Figure 4.6 shows that increasing the number of robots N_R results in faster coverage; robots perform goal layer optimisation and plan paths towards distinct regions. The steady-state values of RMS_ψ decrease as N_R increases, and are significantly lower than the standard deviation of noise $\sigma_\psi = 0.1$ on robots' sensors.

4.5.4 Varying Communications Radius

When two robots communicate they share information about regions of the environment that are within r_C of their current position. This is useful for scalability as it limits the amount of data transferred between them (robots do not need to share information about their whole map). For the two initial conditions 'corner' and 'random' we measure the time taken for full coverage as well as the steady-state value of RMS_ψ after 1000s for representative low, medium and large r_C.

Table 4.1 shows that a large r_C results in faster coverage and lower RMS_ψ as robots are able to process information about distant regions and can benefit from being spread out as in the 'random scenario. For small r_C, the 'corner scenario (see Figure 4.5) performs better – robots begin close together, resulting in more inter-robot communication as compared to the 'random scenario.

Although a larger communication radius is beneficial for the swarm, this effectively increases the amount of data transferred and presents a trade-off for real-world systems.

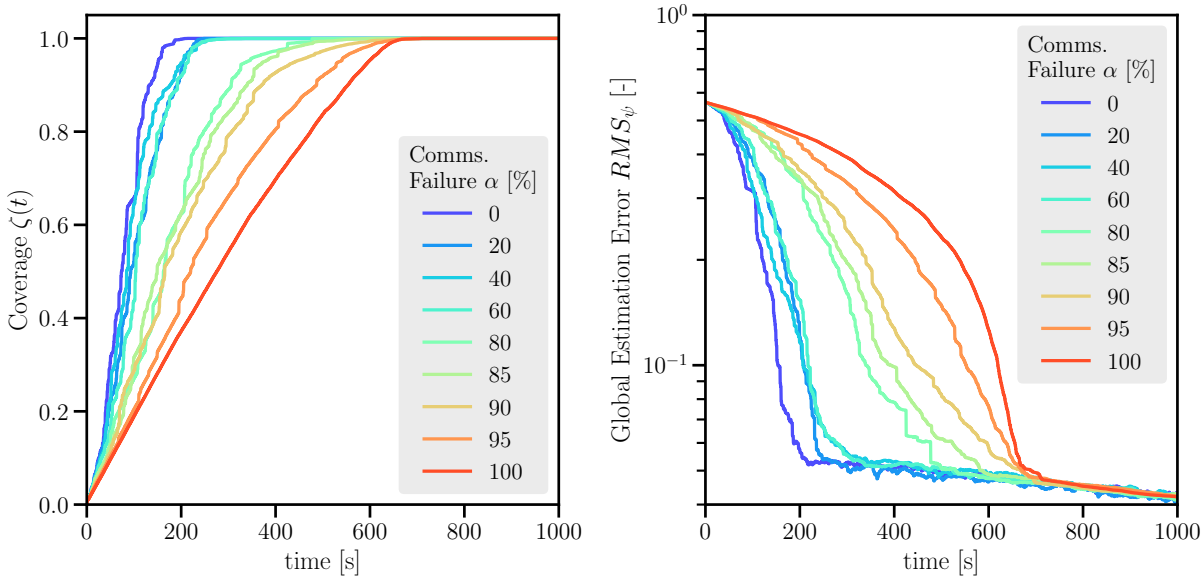


Figure 4.7: Evolution of coverage and RMS_ψ over time as the communications failure rate α varies. Robots take longer to explore the environment as inter-robot communication decreases whilst still accurately estimating the global state.

4.5.5 Performance under Communication Limitations

A key aspect of our multi-layer approach is that robots do not constantly need to take part in information exchange. Real-world robots may be subject to power limitations or sensor failures, and may prioritise collision avoidance over information acquisition. We simulate this regime of communications failure for $N_R = 20$ robots in the 'random' initialisation scenario. At every time step a proportion α of randomly selected robots *do not* take local measurements or perform inter-robot communication in their Goal and Information layers.

Figure 4.7 shows that as α increases it takes longer for the swarm to cover the full environment and to accurately estimate $\mathbf{\Psi}$ as robots cannot efficiently decide on goal locations for path planning, or make use of their neighbours' collected information. Robots continue to optimise over their own GBP stack, making use of new information when it becomes available; full coverage and accurate estimation is still achieved when the robots behave independently ($\alpha = 1$).

4.6 Conclusions

We have shown that our distributed method for collaborative optimisation over multiple problem competencies can enable a team of robots to accurately estimate a global state based purely on local independent measurements. By formulating the problem as a stack of linked factor graphs robots optimise over each problem competence in an asynchronous manner. The distributed nature of our algorithm promises scalability to large environments and numbers of robots.

In future work we will test our algorithm with more complex maps, and investigate optimisation with a dynamic model of the global state as here we have assumed it to be constant with time.

Chapter 5

DANCeRS: A Distributed Algorithm for Negotiating Consensus for Robot Swarms

Negotiation and discussion are the greatest weapons we have for promoting peace and development.

— Nelson Mandela

Contents

The stacked formulation of Chapter 4 addressed optimisation across multiple objectives. However, many tasks in large-scale multi-robot systems fundamentally rely on achieving consensus about some higher level global state. This chapter presents DANCeRS, a distributed GBP-based algorithm that unifies consensus across both continuous parameters and discrete decisions, with applications to swarm formation and collective decision-making.

5.1 Introduction

In nature swarms such as bird murmurations or schools of fish consist of simple agents that communicate locally and exhibit collective behaviour, often requiring agreement on shared global parameters like velocity or orientation. Similarly, robot swarms must act cohesively, whether for task allocation, arrangement, or coordinated behaviours like exploration or aggregation [Brambilla et al., 2013]. This requires each robot to make individual decisions while aligning with the swarm's global objectives.

Centralised systems offer precise coordination by delegating all computation to a single external entity. However, such approaches lack scalability and are prone to single points of failure. In contrast, decentralised systems rely on distributed computation and local communication, enabling agents to share information and iteratively refine their decisions based on their neighbours' states. For discrete decision spaces, agents often share probability distributions over the available choices and update their beliefs until consensus is

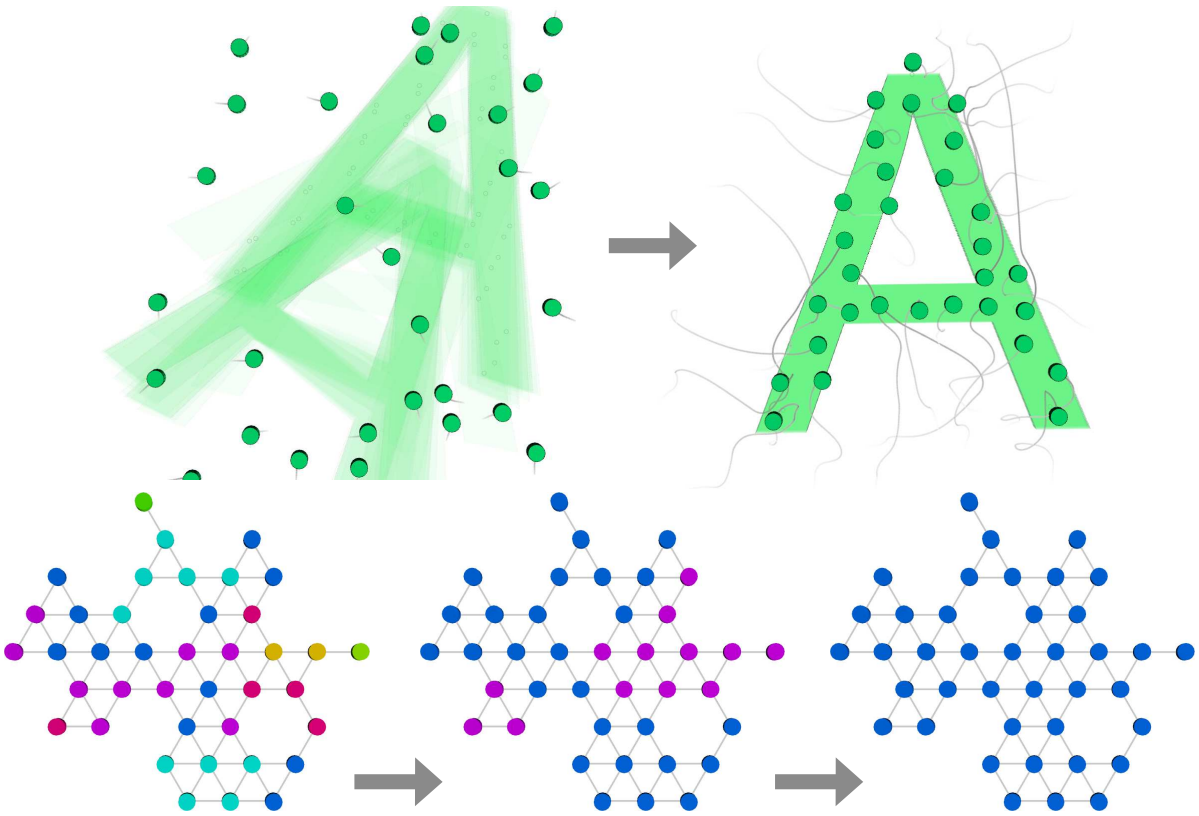


Figure 5.1: (Top) Robots jointly plan paths to arrange into the 'A' shape, achieving consensus over its pose. (Bottom) Robots use the *same* algorithm to achieve consensus over a discrete set of decisions (colours) through local communication.

reached [Zheng and Lee, 2023]. These methods are effective for a swarm deciding between distinct behaviours like exploration [Patwardhan and Davison, 2024], aggregation, or task allocation, commonly framed as best-of-N problems [Valentini et al., 2017].

For continuous spaces, consensus is typically achieved by iteratively updating shared variables, such as positions or orientations, based on differences between neighbouring agents' states. This allows swarms to converge toward global parameters collaboratively, even with limited local communication. Such approaches have been used for shape formation [Sun et al., 2023], where robots must agree on the position and orientation of a shape for coordinated motion. These distributed methods provide scalability and robustness, enabling swarms to function in dynamic environments without centralised control.

Despite significant progress, the literature typically treats consensus over discrete deci-

sion spaces and continuous parameter spaces as separate problems. In this work, we propose a unified framework for achieving consensus in both domains. By modelling the swarm as a sparsely connected factor graph and using Gaussian Belief Propagation (GBP) for distributed inference, our method enables robots to reach consensus efficiently and scalably. This fully distributed approach ensures robustness to increasing swarm size and dynamic environments, addressing both discrete and continuous decision-making in a single cohesive framework.

We demonstrate our general framework in two example applications of joint path planning for shape formation and collective decision-making over a set of discrete options. Our contributions are:

- A unified, distributed framework for consensus in both discrete and continuous decision spaces.
- The extension of GBP path planning for non-holonomic unicycle dynamics in shape formation.
- A novel distributed shape formation method for discrete target assignment, demonstrated with scalability and efficiency in experiments.
- The first application of GBP for discrete consensus in robot swarms.

5.2 Related Work

Achieving global behaviour in robot swarms relies on decision-making frameworks that balance scalability and robustness. While centralised methods are limited by scalability and single points of failure, decentralised approaches enable collective behaviour through local interactions, whilst addressing challenges such as conflict resolution. The literature focuses on two key problems: consensus over (1) a set of discrete options, and (2) over continuous parameter spaces.

In discrete consensus problems, each agent in the swarm must select one option from a finite set based on local communication with its neighbours. This problem, often referred to as the "best-of-N" problem has been widely studied particularly for N=2 options where a swarm estimates the majority value in the environment [Shan and Mostaghim, 2021, Siemensma et al., 2024]. For larger N, probabilistic approaches involve agents maintaining and sharing a probability distribution over the set of options. In [Liu and Lee, 2020] agents update their internal beliefs based on local communication and knowledge of the local consensus group (agents sharing the same decision) whereas in [Zheng and Lee, 2023] agents share decision IDs and levels of certainty of their beliefs based on probabilistic entropy. Other popular distributed approaches include voter methods where robots use information from their neighbours and choose a random decision [Rogers and Gross, 2013], the majority decision [Wessnitzer and Melhuish, 2003] or the most frequent recently observed decision [Scheidler et al., 2016]. Another approach to distributed decision-making is opinion dynamics, which models how agents adjust their opinions through local interactions. Unlike traditional consensus models, it allows for complex social dynamics, including agreement, disagreement, and opinion clustering. The nonlinear model in [Bizyaeva et al., 2023] introduces tunable sensitivity, enabling flexible opinion transitions. While this provides insight into how opinions evolve in networks, such models do not always ensure that all agents reach the same decision, which is often desirable in multi-agent coordination tasks.

Continuous consensus problems involve the swarm negotiating their beliefs about continuous global parameters, such as target locations in search-and-rescue or source-seeking tasks, or the position and orientation of a reference frame for shape formation. Distributed consensus algorithms typically rely on robots effectively averaging their neighbours' beliefs for formation control [Savkin et al., 2016], obstacle avoidance [Afdila et al., 2023] and saliency detection [Alhafnawi et al., 2021]. Recent work applied the mean-shift algorithm to shape formation tasks [Sun et al., 2023], where robots negotiated on the pose of the shape, and were able to form and explore complex patterns defined as

an artificial potential field. This work was extended in [Zhang et al., 2024] to consider formations consisting of distinct points. These mean-shift approaches iteratively adjust robots' velocities to align with local density maxima, enabling collision avoidance and dynamic role assignment, although they are designed for shapes with one continuous connected component. Other approaches, such as local task swapping, have been proposed for distributed shape formation [Wang and Rubenstein, 2020]. In this method, robots iteratively negotiate position assignments by swapping roles, deciding whether to act as anchors or clients based on local priorities. A hop-count strategy propagates goal selection, ensuring collision-free movement in a discretised grid-world. However, this approach assumes robots have already reached consensus on a shared global reference frame. By contrast, we seek to addresses both shape formation and reference frame alignment simultaneously, removing the need for task negotiation based on roles or priority.

Despite being treated as two distinct problems in the literature, real-world tasks often require swarms to achieve consensus in both discrete and continuous decision spaces simultaneously. In our work, we propose a unified framework capable of addressing both types of consensus problems. Our approach builds on Gaussian Belief Propagation (GBP), a distributed algorithm for factor graph inference, which has demonstrated strong performance in dynamically changing graph topologies. GBP is asynchronous, fully distributed, and relies on peer-to-peer message passing between nodes in the graph. Our previous works [Patwardhan and Davison, 2024], [Patwardhan et al., 2023] applied GBP to distributed multi-robot path planning and information aggregation but were limited to Euclidean spaces and holonomic motion models. More recently, GBP has also been used for consensus in 2D spaces [Jones and Hauert, 2024], but without extending to Lie groups, restricting its ability to handle more complex transformations.

In this chapter we introduce a general GBP-based framework for consensus over Lie groups, enabling robots to negotiate over more complex spaces beyond simple Euclidean variables. Unlike [Patwardhan and Davison, 2024], where consensus was achieved through implicit measurement fusion, we formulate it as an explicit negotiation process. Addi-

tionally, we extend GBP to discrete decision spaces, broadening its applicability beyond continuous estimation. We also propose a novel shape formation algorithm, leveraging GBP for distributed coordination, and incorporate a non-holonomic motion model, making our approach more realistic than prior work for practical robotic systems.

5.3 Background

5.3.1 GBP using Lie Theory

Gaussian Belief Propagation (GBP) is commonly used in Euclidean space, where variables are represented as vectors. However, many robotics problems require optimisation over Lie groups, which provide a natural way to represent transformations, orientations, and motions. Lie groups are continuous mathematical structures that generalise Euclidean spaces while preserving properties like smoothness and group operations. They are widely used in robotics to represent states such as positions, rotations, and velocities in a way that respects the underlying geometry.

In consensus problems, robots must agree on shared parameters such as position, orientation, or categorical decisions. While averaging works well for simple Euclidean quantities, it fails for rotations and transformations. For example, in the SO(2) Lie group, averaging two angles near 0° and 360° incorrectly suggests 180°, rather than recognising their proximity. Lie theory allows optimisation to be performed in the correct mathematical space, ensuring accurate consensus.

GBP operates by passing Gaussian-distributed messages between variables in a factor graph. In Euclidean settings, these messages contain a mean vector and a precision matrix, but for Lie groups, they must be transformed into the tangent space of the current belief, updated, and mapped back using exponential and logarithmic maps. This ensures that information is propagated correctly without distorting the underlying geometry. We

have derived the equations for GBP using Lie theory in Chapter 2.4, and direct the reader to [Murai et al., 2024b], [Deray and Solà, 2020] for further insight.

Our method provides a general framework for optimisation on Lie groups, enabling applications in trajectory planning, state estimation, and control. Here, we demonstrate its use for multi-robot consensus over poses and discrete decisions.

5.4 Method

We consider a swarm of N robots moving in $\mathbb{R}^2$ each having a communications radius r_C. At any time $t \geq 0$ they form a sparsely connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{\mathcal{V}_i : i \in [0, ..., N-1]\}$ is the set of robots, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges denoting inter-robot connections. An edge $\mathcal{E}_{ij}$ between robot i and j exists if at time t if $\|\boldsymbol{x}^i - \boldsymbol{x}^j\| < r_C$, where $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ are the 2D positions of robots i and j respectively.

The robots must solve multifaceted problems; they must perform optimisation in path planning and collision avoidance, but also form consensus over some shared global information. Similarly to the approach from [Patwardhan and Davison, 2024], each robot holds a two-layered factor graph stack as shown in Figure 5.2 where each layer of the stack represents a particular aspect of the optimisation problem: the Planning $\mathcal{P}$ and Global Consensus $\mathcal{G}$ layers.

As ours is a purely distributed algorithm, we take inspiration from [Murai et al., 2024b]; each robot maintains a "webpage" of information containing the outgoing messages to the factors and variables in the GBP stacks of each of its connected neighbours. Each robot follows Algorithm 3.

The parametrisation of variables in the Global Consensus layer depends on the specific application to be considered. To demonstrate our general framework which works for many of the common Lie Groups, we perform two types of experiments where the robots

must reach consensus over some shared global information.

In the first experiment, we choose shape formation consensus as a proxy problem where the SE(2) group can be used to represent the position and orientation of a target formation shape. In the second, we examine how the same framework can be used for performing negotiation over the continuous $\mathbb{R}^M$ group, and can be used to perform decision-making and negotiation over a set of discrete options.

We now describe the general form of the two factor graph layers as depicted in Figure 5.2. In general we describe a factor f with its measurement function $\boldsymbol{h}$ and its strength given by vector $\boldsymbol{\sigma}$ such that the measurement precision $\boldsymbol{\Lambda} = diag(\boldsymbol{\sigma}^{-2})$ is a diagonal matrix. As in Chapter 4, a GBP variable is denoted as ${}^{\mathcal{L}}\mathbf{x}_m^R$, where R is the index of the robot that the variable belongs to, $\mathcal{L}$ is the GBP Stack layer, and m is the index of the variable in that layer.

5.4.1 Global Consensus Layer $\mathcal{G}$

In this layer robots may share information about their interpretation of the global parameters and negotiate to collectively form a consensus.

Let χ represent the global parameter that all robots must agree upon. For example, this may represent the index of a shared behaviour to be exhibited (from a list of different behaviours e.g. [*explore, regroup, return-to-home, ...*]), or alternatively a common reference vector to be agreed upon by the robots, such as a common origin. Each robot holds variables ${}^{\mathcal{G}}\mathbf{x} \in \mathcal{M}$ representing the robot's interpretation of the global parameters. In our generalised framework $\mathcal{M}$ may be any one of the common Lie groups such as $\mathbb{R}^M, \mathrm{SO}(2), \mathrm{SO}(3), \mathrm{SE}(2), \mathrm{SE}(3)$.

5.4.1.1 Consensus Domain

A GBP variable ${}^{\mathcal{G}}\mathbf{x}^i$ in this layer represents robot i's belief about the global parameter χ, and consensus is achieved over the continuous domain $\mathcal{M}$.

When applying our method to problems involving consensus over a discrete set of options, we choose $\mathcal{M} = \mathbb{R}^1$ to represent the continuous form of the discrete decision space. We define a quantisation function $\gamma : [0, 1) \to \{0, 1, \ldots, N_D - 1\}$ and its inverse γ^{-1} as

$$\gamma(x) = \lfloor N_D \cdot x \rfloor \tag{5.1}$$

$$\gamma^{-1}(k) = \frac{k}{N_D} \quad \text{for} \quad k = 0, \ldots, N_D - 1 \tag{5.2}$$

respectively where $N_D \in \mathbb{Z}_{>0}$ is the total number of discrete intervals. Transforming the discrete decision space into a continuous space allows the use of GBP for iterative negotiation, enabling the swarm to converge onto a shared common decision. It should be noted that the quantisation function is used simply to extract the robot's discrete decision, and is not part of the consensus algorithm.

5.4.1.2 Prior Factor

Robot i has a prior belief about the global parameter represented by a factor of strength $\boldsymbol{\sigma}_p$ and form

$$f_p : \boldsymbol{h}_p({}^{\mathcal{G}}\mathbf{x}^i) = {}^{\mathcal{G}}\mathbf{x}^i \ \ominus \ {}^{\mathcal{G}}\boldsymbol{x}^{i,0} \ , \tag{5.3}$$

where ${}^{\mathcal{G}}\boldsymbol{x}^{i,0}$ is the initial belief of the formation parameters, and $\ominus$ is the right–minus action on the manifold.

5.4.1.3 Inter-Robot Consensus Factor

When two robots are within communication range of each other, inter-robot factors are created from each variable in this layer. The factors have strength $\boldsymbol{\sigma}_c$ and take the form

$$f_c : \boldsymbol{h}_c({}^{\mathcal{G}}\mathbf{x}^i, {}^{\mathcal{G}}\mathbf{x}^j) = {}^{\mathcal{G}}\mathbf{x}^i \ominus {}^{\mathcal{G}}\mathbf{x}^j = \mathrm{Log}\left(({}^{\mathcal{G}}\mathbf{x}^j)^{-1} \circ {}^{\mathcal{G}}\mathbf{x}^i\right) \ , \tag{5.4}$$

where $\circ$ denotes the composition operator for the Lie Group $\mathcal{M}$.

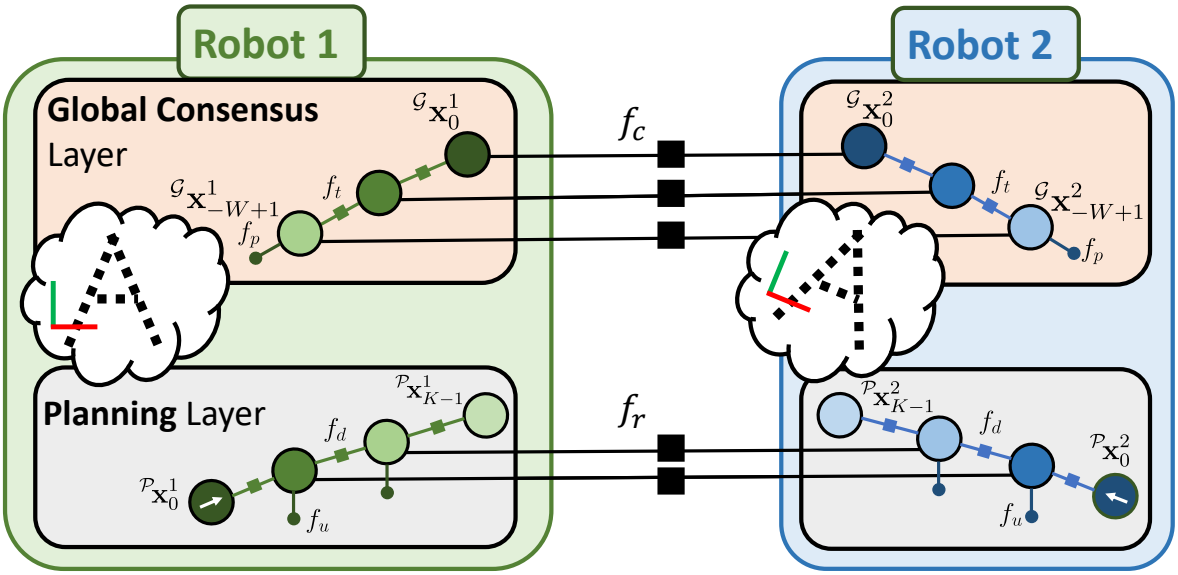


Figure 5.2: Factor graph stack of each robot. Local GBP iterations enable global consensus between the robots, visualised here as the formation parameters for the letter A.

5.4.1.4 Temporal Factors and Variables

The resulting factor graph is dynamic as robots may leave and rejoin a local group at any time. When robots leave a group, they delete their existing inter-robot factors – as GBP variables are memoryless, robots would forget the locally converged belief of the global parameter. Hence, robots maintain a sliding window of W variables ${}^{\mathcal{G}}\mathbf{x}^i_{-k} : k \in [0, W-1]$ in this layer, which are connected with temporal factors $f_t({}^{\mathcal{G}}\mathbf{x}^i_{-k+1}, {}^{\mathcal{G}}\mathbf{x}^i_{-k})$ having the same form as the inter-robot factor in Equation 5.4 with a strength $\boldsymbol{\sigma_t}$. The sliding window is updated every T_S time steps. The oldest variable ${}^{\mathcal{G}}\mathbf{x}^i_{-W+1}$ along with its connected factors

is deleted. A new prior factor f_p is generated at the now oldest variable ${}^{\mathcal{G}}\mathbf{x}^i_{-W+2}$ and initialised with the message from factor $f_t({}^{\mathcal{G}}\mathbf{x}^i_{-W+1}, {}^{\mathcal{G}}\mathbf{x}^i_{-W+2})$. This prior factor represents the marginalised information from the deleted portion of the factor graph.

The resulting effect is that a robot disconnecting from a local group maintains its previous belief mean, although its covariance weakens over the sliding window.

5.4.2 Path Planning Layer $\mathcal{P}$

Optimisation in this layer involves robots planning an efficient path in a short forward time window of K time steps, and creating inter-robot factors between themselves and their neighbours for collision avoidance.

A noise-on-acceleration model is used for the robot dynamics such that the robot state are parametrised as

$$ {}^{\mathcal{P}}\mathbf{x} = [\boldsymbol{p}^\top, \dot{\boldsymbol{p}}^\top]^\top = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]^\top \ . \tag{5.5} $$

We build upon the work presented in Chapter 3 where robots with holonomic dynamics models planned paths towards fixed goals. These paths consisted of the robots' planned states ${}^{\mathcal{P}}\mathbf{x}_k$ for $k \in [0, K-1]$ which were the optimisation variables, connected to each other through dynamics factors f_d representing a constant velocity model constraint. In this work however we propose a new unicycle model factor f_u, enabling modelling of non-holonomic motion, and improvements over the previous work, which we detail here.

5.4.2.1 Unicycle Model Factor

This factor with strength σ_u is added to each variable in the layer and takes the form:

$$ f_u : \boldsymbol{h}_u({}^{\mathcal{P}}\mathbf{x}_k) = \dot{x}\cos(\theta) - \dot{y}\sin(\theta) \ . \tag{5.6} $$

As factors represent cost functions, optimisation drives the value of the measurement function of this factor $\boldsymbol{h}_u({}^{\mathcal{P}}\mathbf{x}_k)$ to 0. This encourages the robot velocity at any time k to be in the direction of the robot's heading.

5.4.2.2 Collision Avoidance Factor

We also introduce a new inter-robot collision avoidance factor between the variables of robot i and j which has strength σ_r and takes the form:

$$f_r : \boldsymbol{h}_r({}^{\mathcal{P}}\mathbf{x}_{k,i}, {}^{\mathcal{P}}\mathbf{x}_{k,j}) = \exp\left\{-\frac{|\boldsymbol{x}_{k,i} - \boldsymbol{x}_{k,j}|}{d_{min}}\right\}, \tag{5.7}$$

where d is the distance between the two robot planned paths at time step k, and d_{min} is a minimal separation distance between the robots. This factor has a smoother measurement function compared to the equivalent factor in Chapter 3, and so a smoother Jacobian.

5.4.2.3 Horizon Update

As per Algorithm 3, at every time step after N_I iterations of GBP the current and horizon variables of the planned path are propagated forward in time. Unlike previous work, we move the horizon state $\boldsymbol{p}_{K-1}$ towards an application-specific goal location $\boldsymbol{g}$ which can be time-varying. First we create the goal vector $\boldsymbol{d}$ as the straight-line vector from the horizon state's position $[\, \boldsymbol{p}_{K-1}[x],\ \boldsymbol{p}_{K-1}[y] \,]^\top$:

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{g}[x] - \boldsymbol{p}_{K-1}[x] \\ \boldsymbol{g}[y] - \boldsymbol{p}_{K-1}[y] \end{bmatrix}. \tag{5.8}$$

Secondly, the goal vector is used to update the velocity of the horizon state $\dot{\boldsymbol{p}}_{K-1}$, capping the linear and angular velocities at v_{max}, and ω_{max}:

$$\dot{\boldsymbol{p}}_{K-1} = \begin{bmatrix} \min\left(v_{\max}, \frac{\|\boldsymbol{d}\|}{\Delta t}\right) \cdot \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|} \\ \min\left(\omega_{\max}, \frac{|\angle \boldsymbol{d} - \boldsymbol{p}_{K-1}[\theta]|}{\Delta t}\right) \cdot \text{sign}(\angle \boldsymbol{d} - \boldsymbol{p}_{K-1}[\theta]) \end{bmatrix}. \tag{5.9}$$

Algorithm 3 For each robot i

Initialise GBP stack layers $\mathcal{P}, \mathcal{G}$
while $t < T_{max}$ **do**
  Let $\mathcal{N}(i) = \{j \mid ||\boldsymbol{p}_0^i - \boldsymbol{p}_0^j|| < r_C\}$ *be the set of robots within the communication radius of* i.
  Let $\mathcal{C}(i)$ *be the set of robots connected to* i.
  for Newly observed robot $j \in \mathcal{N}(i) \backslash \mathcal{C}(i)$ **do**
    Create inter-robot factors f_c, f_r.
  for Out-of-range robot $j \in \mathcal{C}(i) \backslash \mathcal{N}(i)$ **do**
    Delete inter-robot factors f_c, f_r.
  for layer $\mathcal{L}$ in GBP Stack **do**
    Broadcast inter-robot messages.
    Perform N_I iterations of GBP.
  Update ${}^{\mathcal{P}}\mathbf{x}_0$ and ${}^{\mathcal{P}}\mathbf{x}_{K-1}$ by Δt.
  Create new information layer variables ${}^{\mathcal{G}}\mathbf{x}_k$ with temporal factors f_t.

Finally this new velocity is used to update the position of the horizon state $\boldsymbol{p}_{K-1}$ with a simple dynamics model:

$$\boldsymbol{p}_{K-1} \leftarrow \boldsymbol{p}_{K-1} + \dot{\boldsymbol{p}}_{K-1}\Delta t \ . \tag{5.10}$$

If the specific application requires the robots to randomly explore the environment, a new $\boldsymbol{g}$ is selected when the robot's horizon state is sufficiently near it. Alternatively if the swarm is to create a shape formation, $\boldsymbol{g}$ can be selected using a nearest-neighbour search over potential locations using a novel distance-occupancy based heuristic we propose in section 5.5.1.

5.5 Applications and Simulations

5.5.1 Path Planning and Consensus for Shape Formation

In this problem a swarm of robots with a limited communication radius r_C must perform path planning to efficiently arrange themselves into a shape formation. Each robot has knowledge of the shape of the formation but must negotiate on the formation parameters (position and orientation) with the other robots. The shape of the formation is stored as a set of points in $\mathbb{R}^2$ with given minimum spacing r_S, and robots select the nearest

point as the path planning goal $\boldsymbol{g}$ using a heuristic depending on distance and occupancy based on local information.

The shape of the formation is defined as a set of N_F points $q_n = [\ q[x],\ q[y]\ ]^\top \in \mathbb{R}^2$ for $n \in N_F$, known by all robots in a canonical coordinate frame of reference. We consider that there are as many points in the formation as there are robots such that $N_F = N_R$, and at any time a robot i may use its current belief about the formation parameters ${}^{\mathcal{G}}\mathbf{x}^i = \mathrm{Exp}\left([x^i,\ y^i,\ \theta^i]^\top\right)$ to transform a point $\boldsymbol{p}$ in the canonical frame into the global frame with the group action $\boldsymbol{p}' = {}^{\mathcal{G}}\mathbf{x}^i \circ \boldsymbol{p}$. The transformation from global to canonical frame can be done with the group inverse $({}^{\mathcal{G}}\mathbf{x}^i)^{-1}$. In order to demonstrate our generalised framework, we opt to use the SE(2) group to represent the formation parameters, although alternative representations of position and orientation such as the decoupled bundle of $\langle \mathbb{R}^2, \mathrm{SO}(2) \rangle$ could also be used; see [Deray and Solà, 2020] for further details.

5.5.1.1 Occupancy Weighting (OW)

Robots must select points that are unoccupied by other robots, however this is hindered by the limited communications range. In this scenario robots must be able to explore within the shape. For a robot i, the set of 2D formation points (FPs) Q^i is augmented with an extra value τ_n for each point q_n, which we refer to as the 'occupancy weighting' of the point. We denote this augmented set of points as

$$\widetilde{Q^i} = \left\{ \widetilde{q_n^i} = \begin{bmatrix} q_n^i \\ \tau_n^i \end{bmatrix} \ \forall\ q_n^i \in Q^i \right\} . \tag{5.11}$$

Each robot stores $\widetilde{Q^i}$ as a KD-Tree data structure which enables efficient and fast nearest-neighbour searching with Euclidean distance metrics. The occupancy weighting (OW) dimension of an FP $\widetilde{q_n^i}[\tau] = \tau_n^i$ reflects how recently the point was seen as being occupied by a neighbouring robot. Robots follow Algorithm 4 to select an optimal FP for the path planning goal $\boldsymbol{g}$, and can make use of the most recent beliefs about the positions

of neighbouring robots from the path planning layer. For a robot i, if any neighbouring robots are near (within a range r_N) to an FP q_n^i, the OW value of the point is set to an arbitrarily high value of $\tau_n^i = \tau_0$. For the remaining FPs that are within a distance r_C of robot i, their OW values are set as $\tau_n^i = 0$ as there are no neighbours occupying them. For all other FPs, as further information about the occupancy is unavailable their OW values are decremented by 1. This OW 'decay' indicates that they may still be occupied.

5.5.1.2 Goal Selection

Finally the current position of the robot is transformed into the canonical formation frame, and augmented with a value of 0 in the third dimension. Robots can then perform a nearest-neighbour search across all three dimensions of the FPs $\widetilde{Q^i}$, which enables them to prefer points with low OW values. The effect with and without OW is visualised in Figure 5.3. Without OW, a robot becomes stuck oscillating between two FPs as it moves out of communication range with a robot occupying an FP.

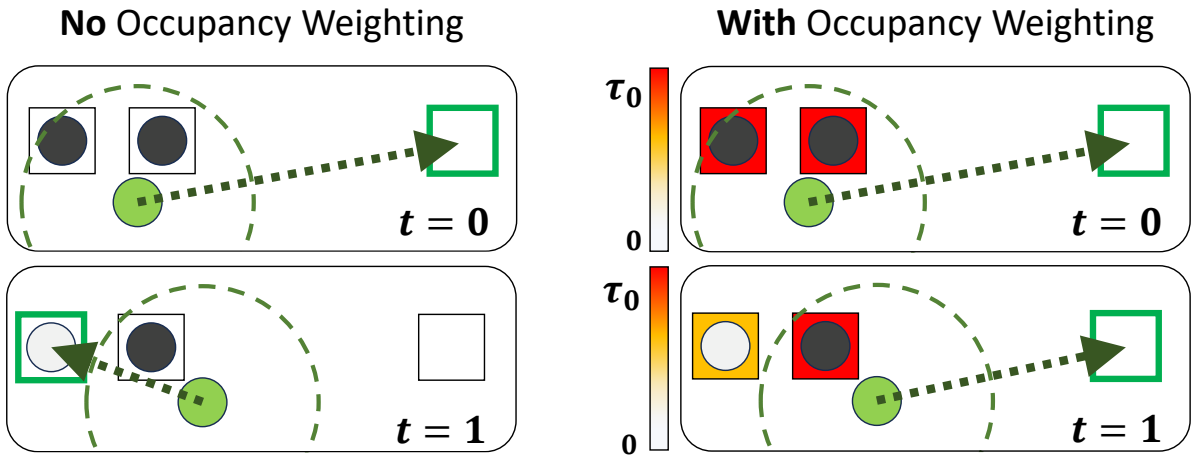


Figure 5.3: Occupancy Weighting (OW): at $t = 0$ the green robot selects the nearest formation point *unoccupied* by its neighbours (gray) to move towards. At $t = 1$ the left-most neighbour is out of communication range and the point it is on appears to be ideal. With OW, the right-most point remains optimal.

5.5.2 Experiments: Shape Formation and Continuous Consensus

Robots of radius 1 m are initialised with random positions and headings $\boldsymbol{p}_i$ in a 100×100 m^2 environment. They follow a non-holonomic unicycle dynamics model, and plan paths with a time window of $T_K = 1.5$ s towards randomly generated goals with a maximum velocity $v_{max} = 2$ m/s and a turning radius $r_T = 2$ m. Additionally we set $\sigma_u = 0.001$, $\sigma_d = 0.1$, $\sigma_r = 0.01$. In all of our experiments in this work we perform *one* iteration of inter-robot message passing per time step only, and set $N_I = 2$. For all experiments in this work we present results over 50 trials and a range of letter formations with $d_{min} = r_N = 2$ m, $r_S = 4$ m, and $\tau_0 = 10^3$.

In the Global Consensus layer, each robot holds a sliding window of $W = 3$ variables, updated every time step ($T_S = 1$). The first of these variables is initialised with a prior factor with observation $\boldsymbol{p}_0^i$; the robot begins by assuming that the formation parameters are the same as its initial position. The strengths of the factors in this layer are set with $\boldsymbol{\sigma_p} = [10, 10, \pi]$, $\boldsymbol{\sigma_s} = 0.1\boldsymbol{\sigma_p}$, and $\boldsymbol{\sigma_r} = 0.01\boldsymbol{\sigma_p}$.

Algorithm 4 For each robot i

for $j \in \mathcal{N}(i)$ **do**
  for $\widetilde{q^i} \in \widetilde{Q^i}$ **do**
    if $\left\| \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \widetilde{q^i} - ({}^{\mathcal{G}}\mathbf{x}^i)^{-1} \circ \boldsymbol{p}_0^j \right\| < r_N$ **then**
      $\widetilde{q^i}[\tau] \leftarrow \tau_0$
for $\widetilde{q^i} \in \widetilde{Q^i}$ **do**
  if $\left\| \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \widetilde{q^i} - ({}^{\mathcal{G}}\mathbf{x}^i)^{-1} \circ \boldsymbol{p}_0^i \right\| < r_C$ **then**
    $\widetilde{q^i}[\tau] \leftarrow 0$
  else
    $\widetilde{q^i}[\tau] \leftarrow \max\left(0,\ \widetilde{q^i}[\tau] - 1\right)$
  $\mathbf{g}_i = X_{i,0}^{\mathcal{G}} \circ \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \left(\underset{\widetilde{q^i} \in \widetilde{Q^i}}{\arg\min} \left\| \widetilde{q^i} - \begin{bmatrix} ({}^{\mathcal{G}}\mathbf{x}^i)^{-1} \circ \boldsymbol{p}_0^i & 0 \end{bmatrix}^\top \right\| \right)$

5.5.2.1 Convergence on Formation Parameters

We vary the radius of communication r_C and the number of robots N_R and measure the number of message passing iterations taken for the swarm to reach a consensus on the formation parameters. We define the convergence criteria to be when the mean inter-robot deviation in the robots' formation parameter beliefs is less than 0.1m in position, and less than 0.01 rad in heading.

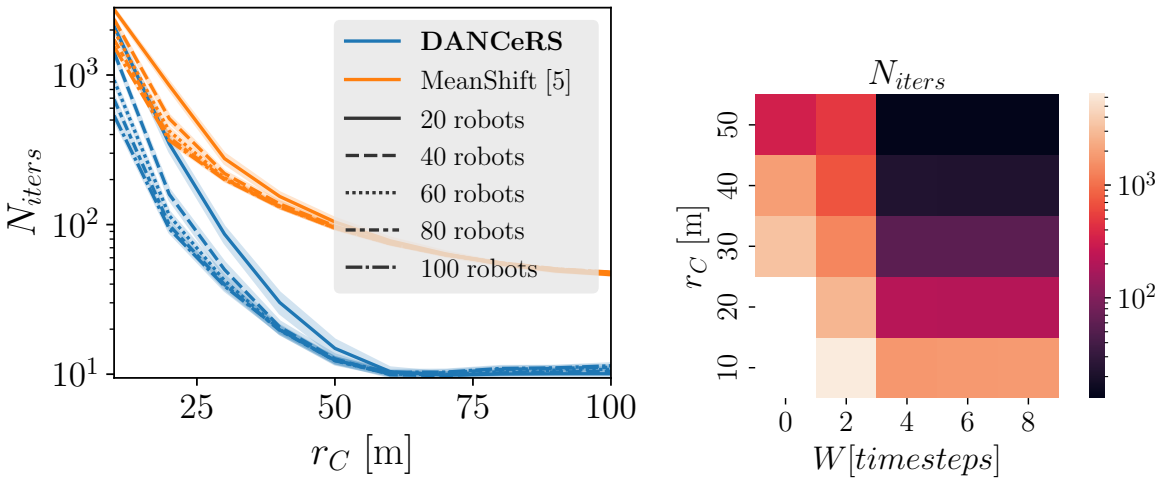
We compare against the distributed consensus algorithm used in [Sun et al., 2023], where agents effectively average their interpretations of formation position and orientation in their local groups. We use the best performing values ($c_1 = c_2 = 1.6$, $\alpha = 0.8$) from the work and consider the scenario of a static formation (stationary with respect to time).

Figure 5.4a shows that we achieve an order of magnitude improvement over the baseline. As r_C increases robots are seen to converge to the global formation parameters in fewer iterations as they are able to exchange information with more robots. This effect is also seen as the number of robots N_R and therefore their density in the environment increases.

5.5.2.2 Effect of Sliding Window on Convergence

We investigate the effect of the length of the sliding window W of variables in the Global Consensus layer on the number of iterations taken until convergence. We consider the case of $N_R = 30$ and vary r_C. Figure 5.4b shows that increasing W results in faster convergence of the formation parameters within the swarm. This trend is more prominent as r_C increases.

When the radius of communication is small, robots forming a local group temporarily negotiate on their beliefs of the formation parameters. However as the variables in GBP are memory-less, when robots leave their local group (altering the factor graph topology) they would lose the previously negotiated belief. Instead using a sliding window of vari-



(a) (b)

Figure 5.4: (a): As the communications radius r_C and number of robots N_R increase, it takes fewer iterations for the swarm to converge onto a consensus on the formation parameters. (b): As sliding window size W and r_C increase, the number of iterations to convergence decreases, shown for $N_R = 30$.

ables, robots are able to maintain their most recent belief due to the new marginalisation factor f_p attached to the now oldest variable. As shown in Figure 5.5, the effect of the sliding window is to maintain the belief of the oldest variable but increase its covariance. With a longer sliding window a robot's belief becomes more susceptible to change when it encounters a new clique of robots with a differing belief.

5.5.2.3 Shape Formation

We present qualitative experiments on robots jointly planning paths to form a pre-defined shape whilst achieving consensus on its position and orientation. The stopping criteria for the experiments is when the distance from each point within the formation to its nearest robot is less than a threshold r_R m. We perform experiments over a range of shapes including those in Figures 5.1, 5.5, 5.6.

Compared to the target-free, mean-shift based shape formation method in [Sun et al., 2023] which is applicable only to shapes consisting of one connected component, in our

method robots explore the shape when they observe that all formation points nearby are occupied by other robots. We are thus able to handle disjoint shapes comprising sparse regions, such as the Smiley Face formation which mean-shift based methods cannot.

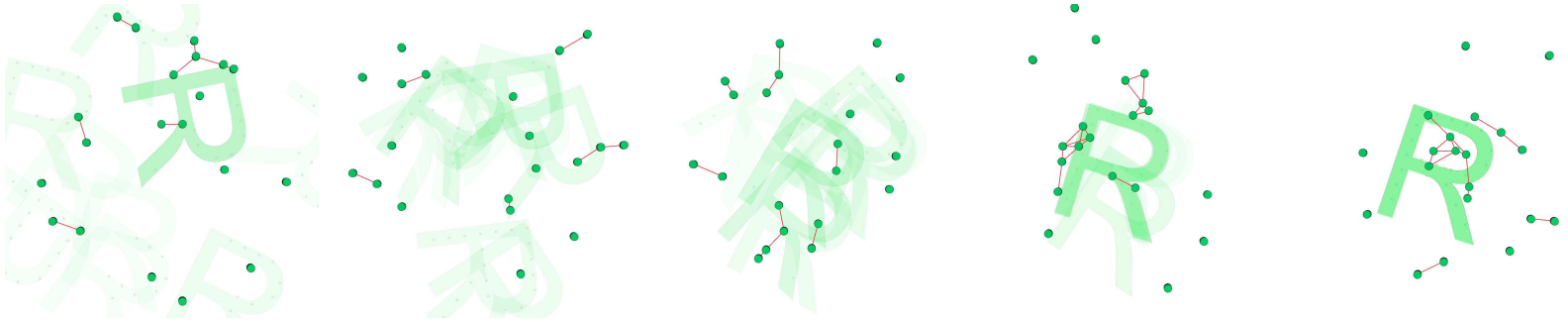


Figure 5.5: Robots with a very small comms. radius r_C move randomly across the environment creating a dynamic graph (red lines show connectivity). Due to the sliding window of Global Consensus variables they are able to gradually form a consensus on the formation parameters (visualised with the shaded letter R).

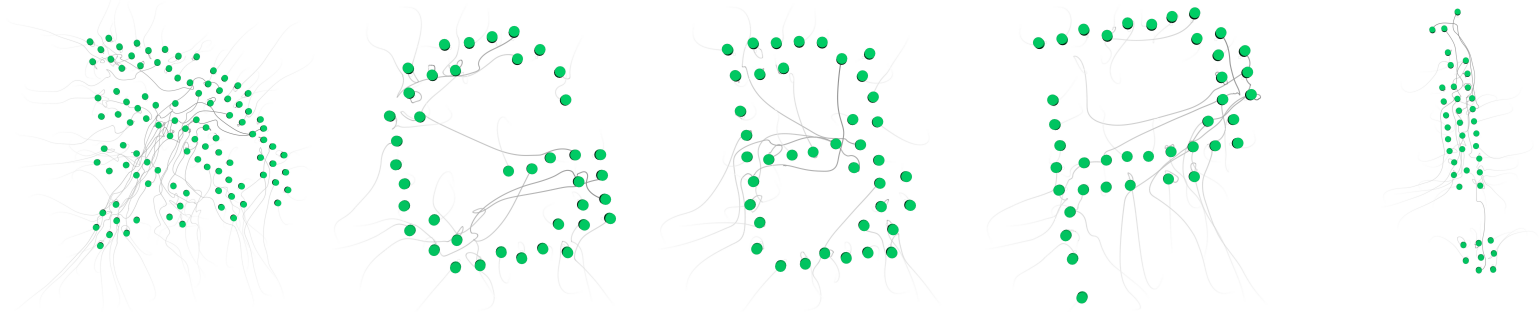
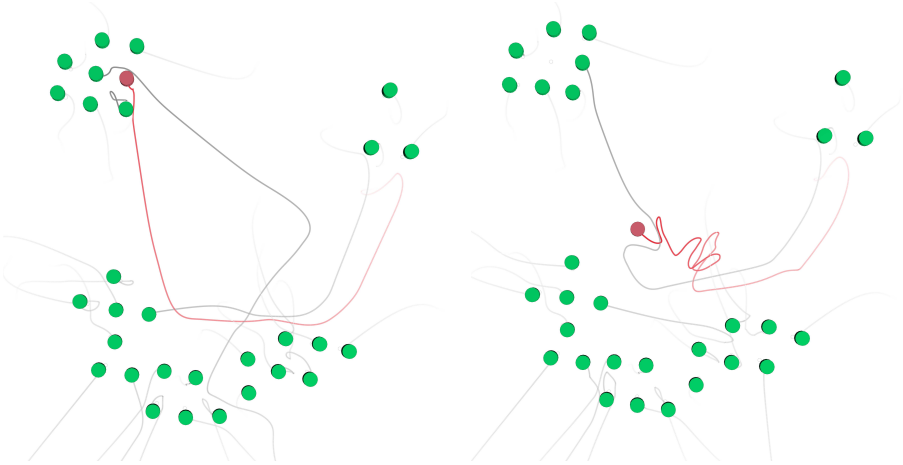


Figure 5.6: Qualitative examples of the shapes our method can enable swarms to create. Robot paths are shown in gray. Joint path planning and shape formation is possible even for shapes with multiple disconnected components including '!' and 'WiFi'.

5.5.2.4 Occupancy Weighting

In Figure 5.7 we show qualitatively the effect with and without occupancy weighting (OW) in the case of the Smiley Face formation ($N_R = 50$, $r_C = 30$ m).

Without OW (Figure 5.7b) and a small r_C, robots are not able to complete shape formation, as they oscillate between choices for goals. With OW however (Figure 5.7a), robots effectively maintain a memory of observed, unoccupied points, encouraging exploration of the rest of the shape.



(a) OW: exploration. (b) No OW: oscillation.

Figure 5.7: (a) Occupancy weighting (OW) encourages shape exploration. (b) Without OW, the final (red) robot oscillates as it 'forgets' occupancy information about a point once out of range of the occupying robots.

5.5.3 Consensus over a Discrete Decision-Space

We now show how the same framework used for the *continuous* consensus problem can be used for reaching consensus over a *discrete* decision space. This problem can be thought of as a variation on the classical *Best-of-N* problem, where a swarm of agents with limited sensing and communication capabilities must reach agreement on a single decision from a discrete set of N_D possible options. This set may represent specific target locations for the

swarm to visit, or simply various behaviours for the swarm to exhibit (e.g. 'aggregation', 'exploration', 'formation'). The critical objective is for all agents to agree on the same choice, as consensus within the swarm is essential for coherent collective action.

5.5.3.1 Global Consensus Layer

We consider that robots only perform optimisation over the Global Consensus layer $\mathcal{G}$ and do not perform path planning. Robots each hold one variable in this layer, randomly initialised with a decision $d \in [0, N_D - 1]$. For a robot i, a unary prior factor f_p with observation $\mathbf{z}_p = \gamma^{-1}(\mathbf{x}^i)$ and strength given by σ_p is connected to the variable ${}^{\mathcal{G}}\mathbf{x}^i$.

5.5.4 Experiments: Discrete Consensus

To validate our algorithm we compare against two recent works which propose distributed solutions to the discrete consensus problem, namely an Entropy based consensus algorithm [Zheng and Lee, 2023] (which we denote as ECA) and a Probabilistic decision-making consensus algorithm (PCA) [Liu and Lee, 2020]. In the former, robots maintain a discrete probability distribution over the N_D options, and exchange with their neighbours their strongest choice, along with the discrete entropy of their probability distribution, representing a certainty about the choice. In the latter, robots exchange not only their most preferred decision, but also a list of other robots exhibiting the same decision as themselves (their 'local consensus group'). The agents then perform internal updating of their states based on the information received from their neighbours. We choose these algorithms as a baseline for comparison as they were seen to outperform classical consensus methods such as majority rule [Wessnitzer and Melhuish, 2003] and k-unamity [Scheidler et al., 2016], and demonstrated impressive scalability. Additionally, we show comparisons against the method in [Sun et al., 2023] as like our method, it is a consensus algorithm operating on a continuous decision space.

It should be noted that in PCA [Liu and Lee, 2020], the authors assume a fixed connectivity in the network of robots. They also assume that when a robot leaves or joins a local consensus group (by having changed its exhibited decision), the next iteration of the algorithm only proceeds after this news has reached all other robots. In a real-world network and with dynamically changing graphs this may not be reasonable and may increase the time taken for the swarm to converge to a decision.

Using the initialisation of robots in the experiments from [Liu and Lee, 2020], robots are placed in a random triangular grid structure as in Figure 5.1 (bottom) such that the distance between any pair of robots is at least 5m. Robots are initialised with random preferred decisions d^i, and r_C is varied in increments of 6m. We define a swarm to have converged to a decision when all agents exhibit the same discrete decision.

5.5.4.1 Discrete Convergence

We measure the number of message passing iterations N_{iters} until convergence as we vary the communications radius r_C and the number of robots N_R. Figure 5.8 shows that as r_C increases it takes fewer iterations for the swarm to converge onto a common decision. For small r_C it takes longer to converge as the swarm size increases.

We plot only the trials that converged within 1000 time steps. Notably when $r_C = 6$ m the ECA method failed to converge, and resulted in many local high-certainty decision groups. At higher r_C, our method demonstrated its scalability, taking a constant number of iterations to converge as N_R increased.

For the PCA algorithm we followed their assumption that changes to local decision groups are propagated throughout the swarm instantaneously, although in real-world experiments this would vastly increase N_{iters} with N_R as the amount of information shared between robots increases over time as the size of the dominant decision group increases.

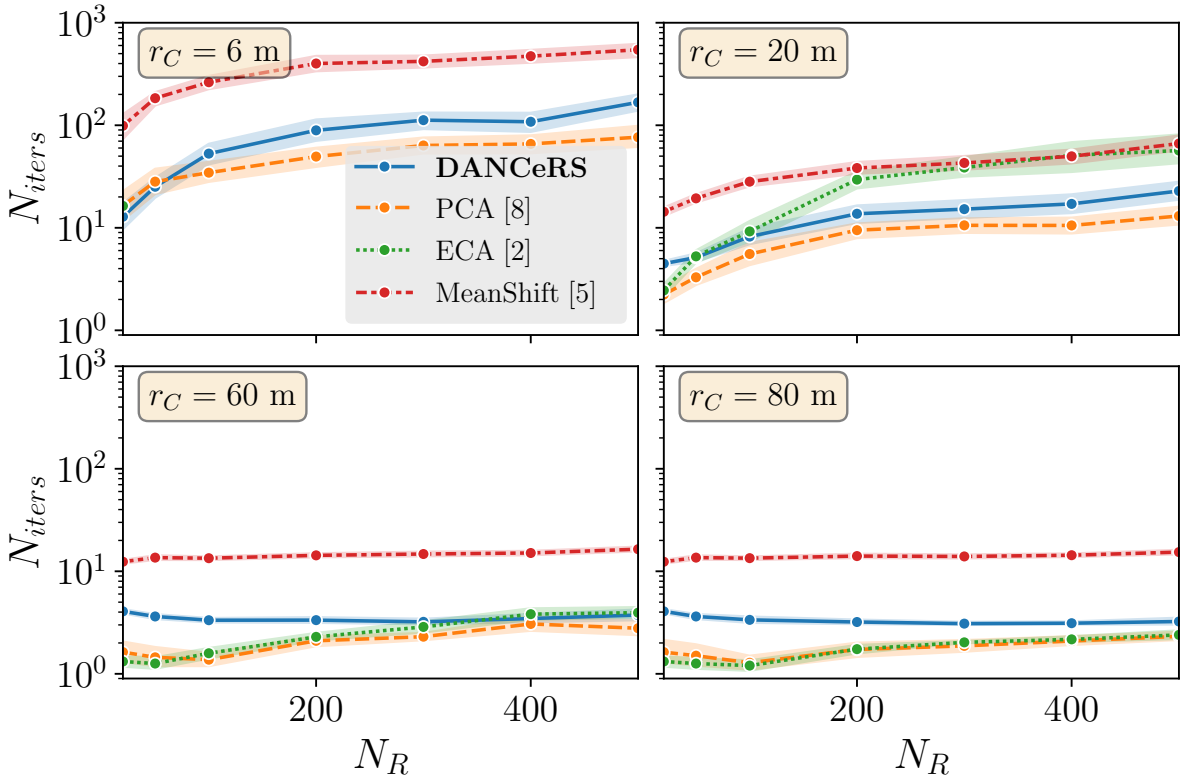


Figure 5.8: Number of iterations N_{iters} taken for the swarm to converge to the same discrete decision as communication radius r_C and number of robots N_R vary.

5.5.4.2 Effect of Sigma Parameter Sweep

We perform an ablation over the strength σ_c of the inter-robot consensus factors f_c and the number of discrete decisions N_D. In GBP the 'strength' of a factor reflects the extent to which a variable's belief may change during optimisation. We hypothesise that an ideal value of σ_c is $\frac{0.5}{N_D}$, as the γ function partitions the continuous domain $[0, 1]$ into N_D discrete bins.

In Figure 5.9 the blue graphs show how the choice of σ_c and N_D affect the time to convergence, with the proportion of successful trials (within 1000 time steps) shown in red. Larger values of σ_c relate to weaker inter-robot consensus factors, leading to fewer successful trials. The vertical blue dotted lines at $\sigma = \frac{0.5}{N_D}$ indicate a favourable upper bound for the factor strength, where there were very few trials that did not converge.

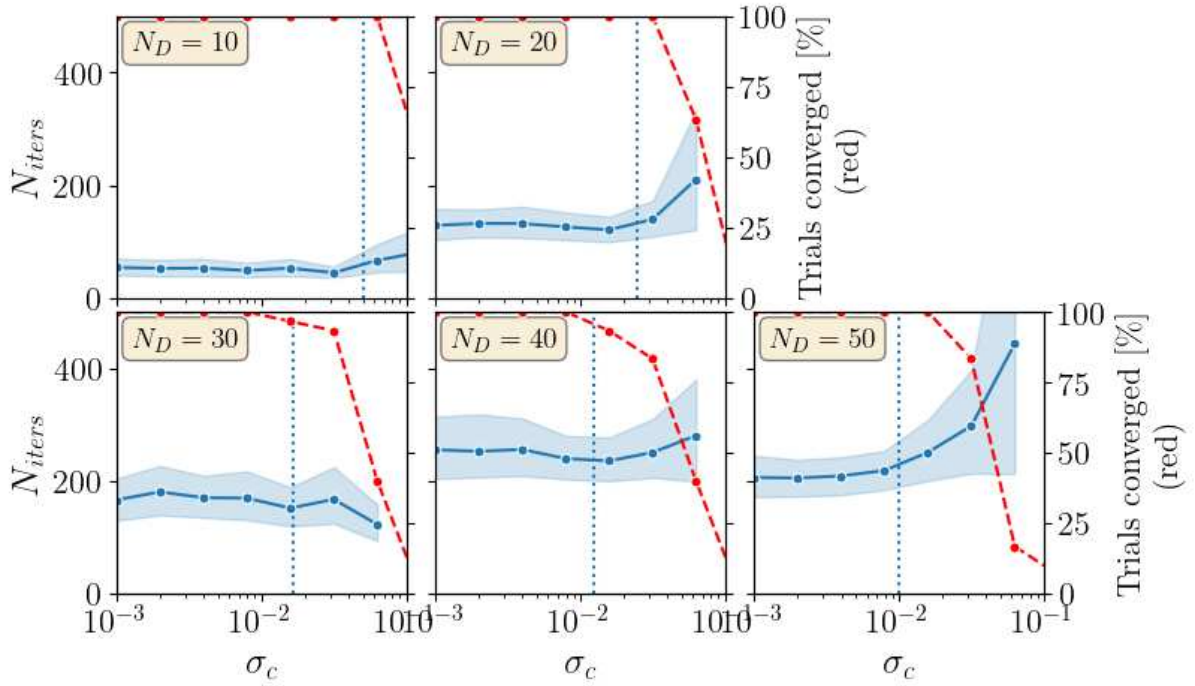


Figure 5.9: Effect of consensus factor strength σ_c and number of decisions N_D on convergence (blue), and % of total trials that converged (red). Vertical dashed lines show $\sigma_c = \frac{0.5}{N_D}$.

Smaller values of σ_c would result in stronger beliefs after convergence, and would not be suitable for problems where the global consensus parameter is time-varying; this is a promising avenue for future work.

5.5.4.3 Informed Robots

A proportion of the swarm may be 'informed' agents, or may have access to salient information about the decision to exhibit. Alternatively it may be desirable to control the large swarm through a small number of agents. We vary the proportion ζ of such 'seed' robots, which all exhibit the same decision and measure the proportion of the swarm that converges to the same decision within 1000 time steps. We consider the case of $N_R = 500$ and the difficult case of $r_C = 6$ m such that each robot in the triangular grid is in communication with its immediate neighbours only.

For our method, we use a strong prior factor on the robot's Global Consensus layer variable with $\sigma_p = 10^{-10}$. Table 5.1 shows that our method resulted in convergence of the swarm to the seed robot decision throughout the range of seed robot proportions considered, from $\zeta = 0.002$ to $\zeta = 0.1$ (corresponding to 1 and 50 seed robots respectively).

Table 5.1: % Trials converged as proportion of seed robots ζ varies for $N_R = 500$ and $r_C = 6$ m.

ζ	0.002	0.01	0.02	0.05	0.10	0.15	0.20
ECA [Zheng and Lee, 2023]	0	0	0	0	0	0	0
PCA [Liu and Lee, 2020]	9	18	31	94	100	100	100
DANCeRS	80	100	100	100	100	100	100

5.6 Conclusions

We have presented DANCeRS, a distributed algorithm for achieving consensus in robot swarms using Gaussian Belief Propagation (GBP). By modelling swarm coordination as a factor graph, we enable scalable and decentralised decision-making in both continuous and discrete domains, relying only on peer-to-peer communication.

We demonstrated its flexibility through two proxy applications: joint shape formation and path planning, as well as collective decision-making over discrete options, showcasing its ability to handle both types of consensus problems. The lightweight and fully distributed nature of GBP means that our method is well-suited for real-world devices with low power and compute; each robot is only responsible for optimisation over its local portion of the global factor graph, and inter-robot messages from each variable of a robot's GBP stack consist of an n-vector and an $n \times n$ symmetric covariance matrix, with n=3 for shape formation and n=1 for discrete consensus.

Some limitations are that robots must store all points in a static formation, which can be memory-intensive for large shapes, and that discrete consensus assumes a fixed set

of options, limiting adaptability to dynamic decision spaces. Despite these, DANCeRS provides a scalable, distributed solution for real-world multi-robot coordination, enabling robots to jointly negotiate continuous and discrete decisions in dynamic environments.

Chapter 6

A Fully Distributed Real-World Implementation of Gaussian Belief Propagation

Science is about knowing; engineering is about doing.

— Henry Petroski

Contents

The rise of distributed multi-agent systems and the Internet of Things (IoT) has led to increased interest in algorithms that can operate without central coordination. A core motivation behind these systems is robustness: removing the single point of failure inherent in centralised control brings about scalability more naturally and allows systems to adapt to uncertainties present in the real world.

Despite this, many impressive demonstrations of multi-robot coordination that claim to be "distributed" still rely on centralised infrastructure for communication such as WiFi access points or 4G networks. Examples include large scale multi-robot formation using local task swapping [Wang and Rubenstein, 2020], the coordinated multi-robot localisation experiments in [Murai et al., 2024b], and the gas sensing and source seeking work in [Duisterhof et al., 2021]. Although the experiments in these works show the potential of scalable distributed coordination, they rely on the presence of a centralised communications network.

Another limitation of these approaches is cost and complexity. Many research platforms are built with specialised hardware and networks that are difficult to reproduce outside of controlled laboratory conditions. For distributed multi-robot methods to become practical at scale, the underlying communication and computation should be lightweight, cheap and easily integrated into existing robotic platforms. Ideally, distributed coordination should be as simple as attaching a low-cost module that enables robots to exchange information directly with their neighbours, without any additional infrastructure.

In this chapter, we present the first real-world demonstration of fully decentralised Gaussian Belief Propagation (GBP) executed on low-cost hardware. Each agent operates independently, communicates using peer-to-peer wireless protocols, and collectively performs distributed inference. This setup enables scalable, robust, and asynchronous consensus without relying on any infrastructure, making distributed coordination practical and accessible.

6.1 System Overview

We define a **device** as a single system capable of performing optimisation and message passing. A device in this case consists of two components:

- A **processor**, running the core GBP algorithm in Python. This could be a Linux machine, a Raspberry Pi variant or any other Single Board Computer (SBC).

- An **microcontroller**, acting as a transceiver responsible for handling transmitting and receiving messages, polling for new devices, and interfacing with the processor. Any microcontroller and communications protocol could be used in theory, as long as peer-to-peer direct communication is possible without the need for a centralised network, or pre-defined leader/follower roles between devices. We use the ESP32

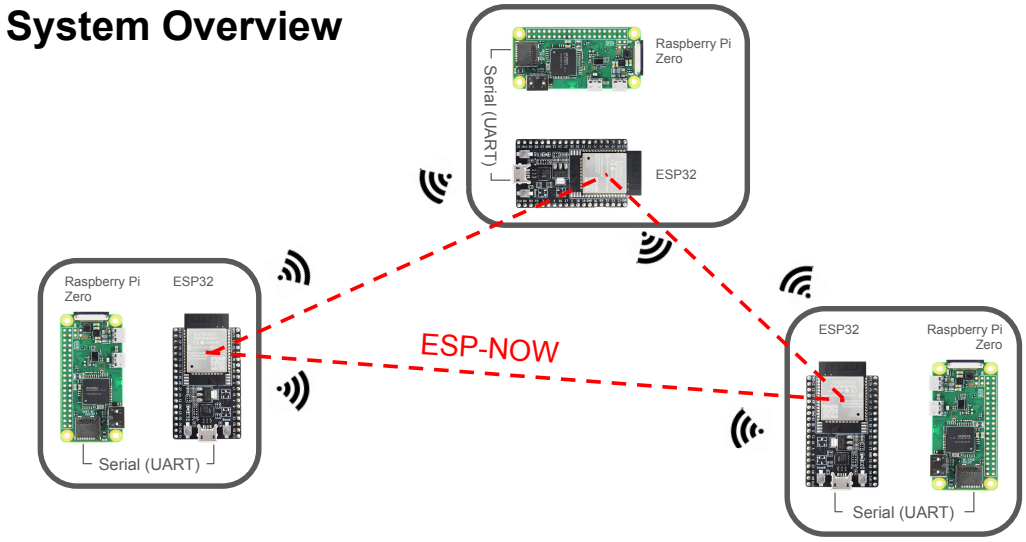


Figure 6.1: System architecture showing 3 **devices** (robots) which each consist of a Raspberry Pi processor connected to an ESP32 microcontroller via UART. The GBP algorithm is run on the processor, and the role of the ESP32 microcontroller is largely to act as a transceiver; inter-robot messages are transmitted and received using the ESP-NOW protocol.

microcontroller which uses the proprietary ESP-NOW protocol [Espressif, 2025], capable of transmitting 250 bytes at a time, peer-to-peer.

These components communicate with each other via a serial UART link. The system overview is shown in Figure 6.1.

6.2 Processor (Raspberry Pi)

Up to now the work in this thesis has involved code written in C++, which is efficient, fast, and can simulate multi-agent behaviour. However since the aim for this chapter is to

interface with a microcontroller and to apply GBP to standard devices used in robotics (motors, actuators, LEDs, robots), we choose to use Python for the GBP algorithm as it can then easily be added to any existing robotic system. Although any Single Board Computer (SBC) could be used to run the Python code, we choose to use the Raspberry Pi 3B+ due to its ease of availability and low cost. Since the core GBP algorithm is of the order of a few hundred lines of code, and is not very demanding in terms of compute, an attractive alternative to the Raspberry Pi 3B+ is the Raspberry Pi Zero W, which is even lower cost.

We create a main Python script which creates, stores and updates a factor graph, but also interfaces with the ESP32 microcontroller. There are some modifications that must be made to the existing C++ GBP implementation to take account of message passing to external devices, as well as minimising communications overhead during the creation and deletion of nodes in the multi-device global network.

6.2.1 Data Structures for Variable and Factor Nodes

Our GBP implementation in C++ [Patwardhan, 2023] serves as the source code for Chapter 3 and the basis for the code for Chapters 4 and 5. Fundamentally we represent a factor graph as a set of variable and factor nodes. The data structure for each type of node has four important concepts or components: a **key** to identify the node, a list of **connected nodes**, an **inbox**, and an **outbox** which we will discuss briefly.

Key. Each node can be identified with a **key** containing information about the device or robot it belongs to, the index of the GBP Stack layer, an individual node id, as well as a flag to indicate the type of node it is (variable or factor):

$$\texttt{node key} = \texttt{robot id} \,\texttt{layer id} \,\texttt{node id} \,\texttt{type} \,. \tag{6.1}$$

As each field is stored as 1 byte, the total data size of a key is therefore 4 bytes.

Connected Nodes list. Each node has a list containing the keys of all connected nodes. Outgoing GBP messages are only created for the nodes in this list. The act of *registering* a new node consists of adding the new node key to this list, as well as initialising entries in the inbox and outbox of the nodes.

Inbox and Outboxes. The node *assumes* that there are messages ready to be processed in the **inbox**. After processing (performing the GBP update steps), outgoing messages are created for each node in the list of connected nodes, and they are stored in the **outbox**. As such, the nodes are fully abstracted from the actual transmission process: they do not need to handle *how* messages are exchanged with external devices.

The structure of a message between two nodes (which we define as a **Node2Node message**) is:

$$\texttt{Node2Node} = \texttt{DoF} \;\; \texttt{from node} \;\; \texttt{to node} \;\; \texttt{belief} \tag{6.2}$$

$$\texttt{belief} = \begin{bmatrix} \boldsymbol{\eta}^\top & \text{ser}(\boldsymbol{\Lambda}) & \boldsymbol{\mu}^\top \end{bmatrix} \tag{6.3}$$

The message contains the keys of the sender and receiver nodes, as well as the GBP belief parameters of information vector $\boldsymbol{\eta}$, precision matrix $\boldsymbol{\Lambda}$, and mean vector $\boldsymbol{\mu}$. We can serialise the whole message into a byte stream by making use of the fact that the precision matrix is symmetric and can therefore be minimally represented by flattening its upper triangular elements. The ser$(\cdot)$ function represents this operation. The byte stream is to be parsed sequentially by the microcontroller, and may contain messages of variables from different Lie Groups. We therefore prepend the serialised message with the degrees of freedom (DoF) of the Lie Group variable in question, from which the total length of message can be inferred.

For a message to be sent from variable $\mathbf{x}_i^A$ to factor f_j^A, since both of these nodes exist on the same device message 'transmission' can happen instantaneously; the message is simply moved from the memory location of $\mathbf{x}_i^A$'s outbox to that of f_j^A's inbox, as depicted in Figure 6.2.

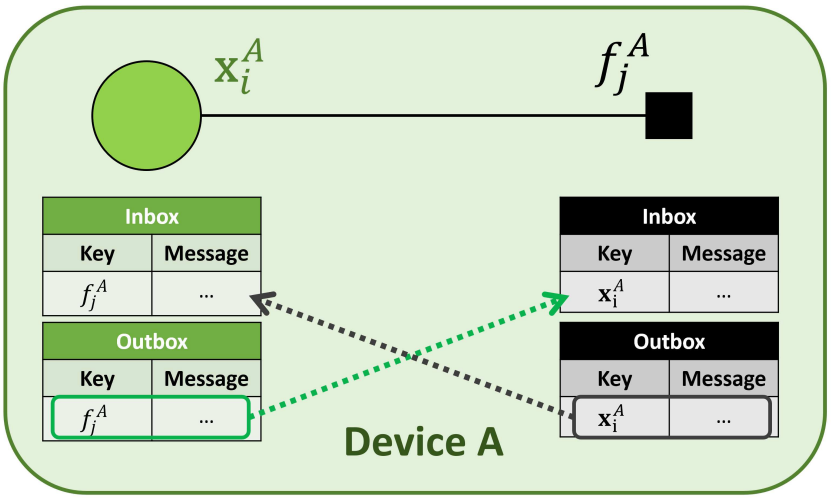


Figure 6.2: Schematic showing transmission of Node2Node messages when both nodes belong to the same device A. The message is simply moved between the appropriate inboxes and outboxes.

6.2.2 External Mailboxes

However, consider the case where the two nodes belong to different physical devices. In this case, the message from variable $\mathbf{x}_i^A$ of device A to factor f_k^B of device B must physically be transmitted to device B, which does not happen instantaneously. In this chapter we therefore introduce **external** inboxes and outboxes acting as intermediaries in the process.

Variable $\mathbf{x}_i^A$ already has factor f_k^B's key in its list of connected nodes. Once $\mathbf{x}_i^A$ has created the outgoing message to factor f_k^B, the message is moved from $\mathbf{x}_i^A$'s outbox to device A's external outbox, ready to be physically transmitted to device B. The transmission step is handled by the ESP32 microcontroller.

Conversely, an incoming message from factor f_k^B of device B to variable $\mathbf{x}_i^A$ of device A first arrives at device A's external inbox. After a triaging process it is then moved to $\mathbf{x}_i^A$'s inbox. This is shown in Figure 6.3.

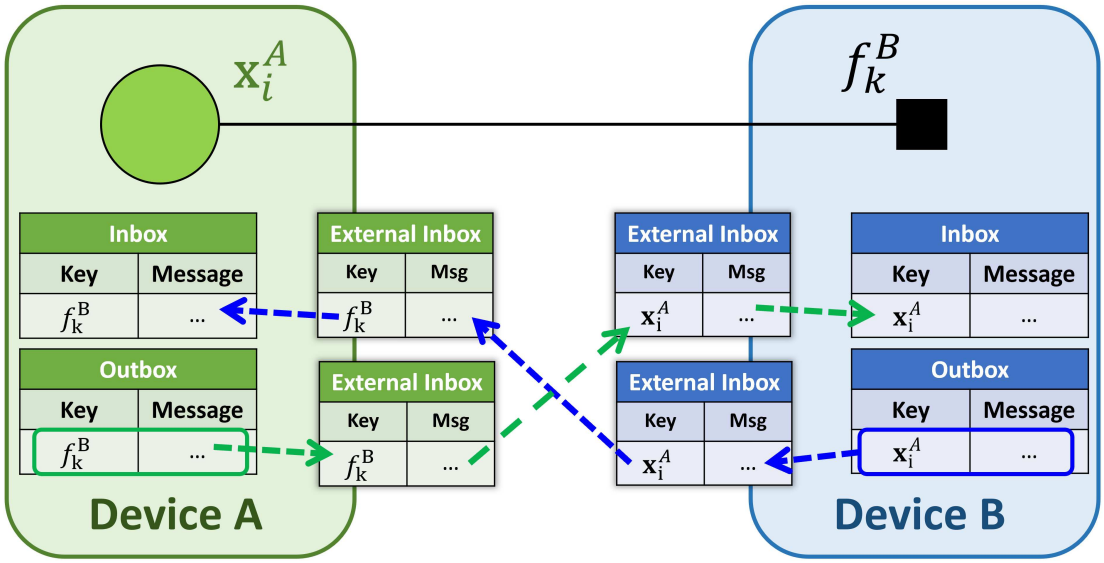


Figure 6.3: Schematic showing message transmission between a variable node $\mathbf{x}_i^A$ in device A and a factor node f_k^B in device B. Each node reads from and writes messages to its (internal) inbox and outbox respectively. If the outgoing message is for a different device's node, the message is first transferred into the external outbox. An incoming message from another device's node is first received in the external inbox, and then transferred to the inbox of the recipient node after triaging.

6.2.3 Triage of Received Messages

Each message from the external inbox of the device is moved to the inbox of the appropriate recipient node in the factor graph *if the sender node is in the recipient node's list of connected nodes*. Depending on the problem, new nodes may be created, or existing nodes may be destroyed. For instance, in Chapter 5 robots create and delete variables and factors as they update their sliding window of variables. In either case, this information must reach external devices. The benefit of triaging with an external inbox is that we can obtain information about newly created or destroyed nodes without needing to send extra information.

New external node. Consider that device B has just created a new factor f_k^B to variable $\mathbf{x}_i^A$ of device A. Naturally, $\mathbf{x}_i^A$ does not yet have f_i^B in its list of connected nodes.

When the factor-to-variable message from device B is received by device A, through the triage process it becomes apparent that this is a new factor (with respect to $\mathbf{x}_i^A$). The variable therefore *registers* the new factor.

Deleted external node. If device B deletes an external node f_k^B which was previously connected to and registered with variable $\mathbf{x}_i^A$ of device A, device B send a factor-to-variable message containing an **empty belief** (length 0). This is a sufficient indicator for device A to *unregister* device B's factor from variable $\mathbf{x}_i^A$.

6.2.4 Processor → ESP32 Interaction

The processor communicates with the ESP32 via serial UART over USB. As such, the processor sends a command of at least 4 bytes, and reads the reply. There are four types of command:

Neighbours: `N` `X` `0` `0`

The processor is requesting a list of nearby active devices.

Receive: `R` `X` `0` `0`

The processor is requesting all *complete* messages received from other devices.

Transmit: `T` `X` `b` `b`

The processor wishes to send data to external devices. The ESP32 should read the next `b` `b` bytes of data and parse it into per-robot messages.

Reset: `X` `X` `0` `0`

The processor is requesting that the ESP32 resets.

Initialise: `0` `X` `0` `0`

The processor is commanding the ESP32 to activate and begin polling.

Setting: `S` `X` `b` `b`

The processor is requesting that the ESP32 changes its settings with the next `b` `b` bytes.

6.3 ESP32 Microcontroller

Any variant of the ESP32 can be used provided it can follow the ESP-NOW protocol. This is a wireless communication protocol developed by Espressif [Espressif, 2025] and originally intended for use in low-power control of smart devices. It is similar to the low power 2.4Ghz WiFi protocol, and replaces the first five layers of the OSI model of communication, reducing communications overhead and increasing the data rate.

The general operation of the ESP32 in this implementation is that it waits for a serial message command from the processor, performs the requested command, and returns a serial message to the processor. Initially, after receiving the **Initialise** and **Setting** commands from the processor, it *becomes active*, with device ID given by the processor.

6.3.1 Polling and Neighbours

Once active, the ESP32 broadcasts a polling message `P` `X` `device id` consisting of its own device ID at regular time intervals of t_{poll} s, which any nearby device following the ESP -NOW protocol receives.

6.3.1.1 ESP32 to Processor Data Transfer

On receiving a poll message from another device, it adds the new device to its list of neighbours if it doesn't already exist. A robot is removed from the list if no poll message has been received for a time threshold t_{forget}. On receiving a **Neighbours** command from the processor with code `N` `X` `0` `0`, the ESP32 sends a list of current neighbour device IDs as a serial byte message:

ESP32 → Processor:

`N` `X` `b` `b` `device id 1` `device id 2` `...` `device id n` , (6.4)

where each coloured box is 1 byte, and `b` `b` is the length of the message payload (in this case equal to the total number of neighbours `n`) in bytes.

6.3.2 Transmitting Messages

On receiving a **Transmit** command `T` `X` `b` `b` from the processor, the ESP32 reads and parses the next `b` `b` bytes of data. This consists of multiple sequential byte streams to transmit, one to each external device. The data sent from the processor to the ESP32 is of the form:

`T` `X` `b` `b` `Device Message 1` `Device Message 2` `...` , (6.5)

where we define a **Device Message** as:

`Device Message n` = `LEN` `GTH` `device id` `Node2Node 1` `...` , (6.6)

and `device id` represents the desired recipient of this message. Note that each Device Message can contain multiple Node2Node messages which were defined in Equation 6.2. Since all data is parsed sequentially, a Device Message begins with the length of data `LEN` `GTH` to be parsed.

Message Chunking. The ESP-NOW protocol has an maximum payload size of 250 bytes that it can transmit. However depending on the number of nodes in the GBP problem, the full message to any external device may exceed this amount. We therefore split the full message into chunks if necessary, before sending one chunk at a time. The message that is physically transmitted to the external device consists of the full message, along with meta-data about the message chunk that is being sent. The maximum length of a chunk is therefore 247 bytes, and the physical device message has the structure:

```
Physical Message {
    current_chunk_number    // 1 byte
    total_num_chunks        // 1 byte
    chunk_size              // 1 byte
    Device Message          // max 247 bytes
}
```

6.3.3 Receiving Messages

The ESP-NOW protocol uses a callback function that runs on receipt of a physical message over the radio. The ESP-NOW documentation highlights that this callback function should not have heavy processing to do otherwise it has an effect on the overall processing capability of the ESP32 device. The case of receiving a polling message has already been covered in Section 6.3.1. Otherwise, the received message is used to update an `inbox` (stored locally on the ESP32), which contains the messages received from other devices.

To account for messages appearing out of order or being lost due to wireless issues, we ensure that a new message chunk i from device d is only appended to the inbox if the last received chunk from device d was index $i - 1$. If the new message has chunk index 0, it takes priority and *replaces* any partially formed message from device d in the inbox. If it was the *final* chunk, the whole completed message is transferred to a separate

`completed_messages` data structure also stored locally on the ESP32.

6.3.3.1 ESP32 to Processor Data Transfer

When the ESP32 receives a **Receive** command `R` `X` `0` `0` from the processor, all messages in `completed_messages` are sent as a serial byte stream to the processor:

`R` `X` `b` `b` `Device Message 1` `Device Message 2` `...` , (6.7)

where each `b` `b` is the length in bytes of the message payload to parse, consisting of multiple Device Messages. Each Device Message is of the form of Equation 6.6, except that in this case the `device id` field indicates the *sender* device.

6.4 Experiments

We demonstrate multi-agent GBP optimisation using such real-world devices by formulating a distributed consensus problem on the manifold, similar to that considered in Chapter 5. The aim is to show a *qualitative* proof-of-concept, showing that GBP can indeed be deployed in practical settings.

In this experiment, we initialise $N_R = 5$ devices, each with N_V variables defined on the SO(2) Lie group. The choice of Lie group is arbitrary; it serves as a simple example with one degree of freedom, though any other group could have been used. The corresponding factor graph is shown in Figure 6.4 for the case of $N_R = 2$ and $N_V = 3$. Each device contains N_V variables, where every variable is associated with a unary factor f_p acting as a prior, initialised through the factor's observation $\mathbf{z}_p$. Inter-device consensus factors f_c are then established across the robots' variables, enforcing the constraint that each robot should reach agreement on the value of every variable (e.g., variable k of each robot should converge to the same value).

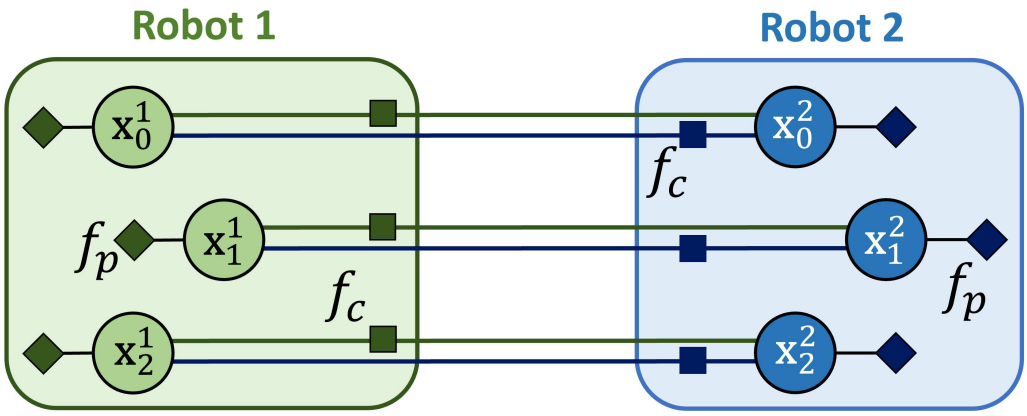


Figure 6.4: Example factor graph for the consensus experiments in this chapter. Shown here is the general case with $N_R = 2$ devices, each with $N_V = 3$ variables (circles), each with unary prior factors f_p (diamonds). Consensus factors f_c (squares) connect corresponding variables between the devices.

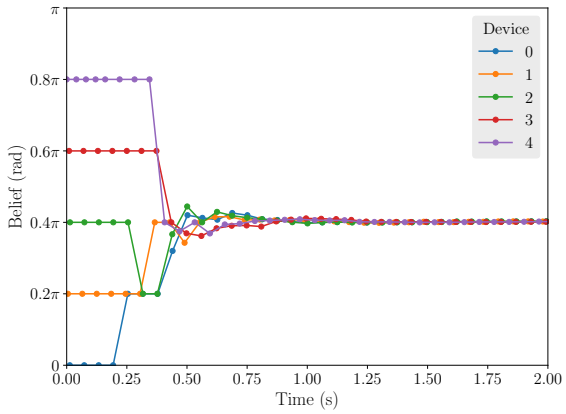
Once activated:

- Each device autonomously discovers its neighbours via polling.
- Messages are exchanged, and GBP is executed locally.
- Despite asynchronous communication over a real radio channel, all devices successfully converge to a common belief.

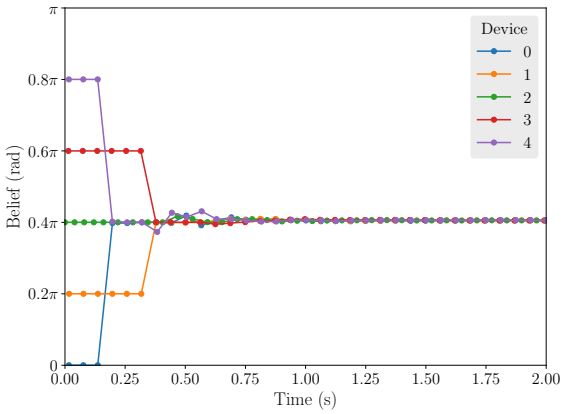
6.4.1 Single Variable Consensus

For the case of $N_R = 5$ devices, each with $N_V = 1$ variable, we track the evolution of the belief means as the devices converge towards consensus on the group average. Since each device d maintains only a single variable $\mathbf{x}_0^d$, the terms 'device belief' and 'variable belief' are used interchangeably. The belief mean for device d is initialised by setting the observation $\mathbf{z}_p^d = \frac{d}{N_R}\pi$. Consequently, the expected group average is:

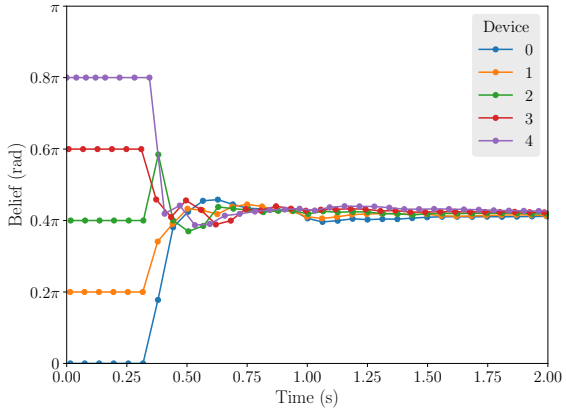
$$\frac{1}{N_R}\sum_{d=0}^{N_R-1}\frac{d}{N_R}\pi = 0.4\pi \ . \tag{6.8}$$



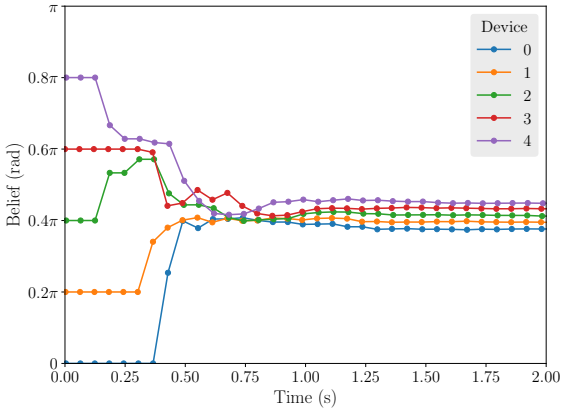
(a) $N_V = 1,\ \sigma_c = 0.01$



(b) $N_V = 1,\ \sigma_c = 0.1$



(c) $N_V = 1,\ \sigma_c = 0.5$



(d) $N_V = 1,\ \sigma_c = 1.0$

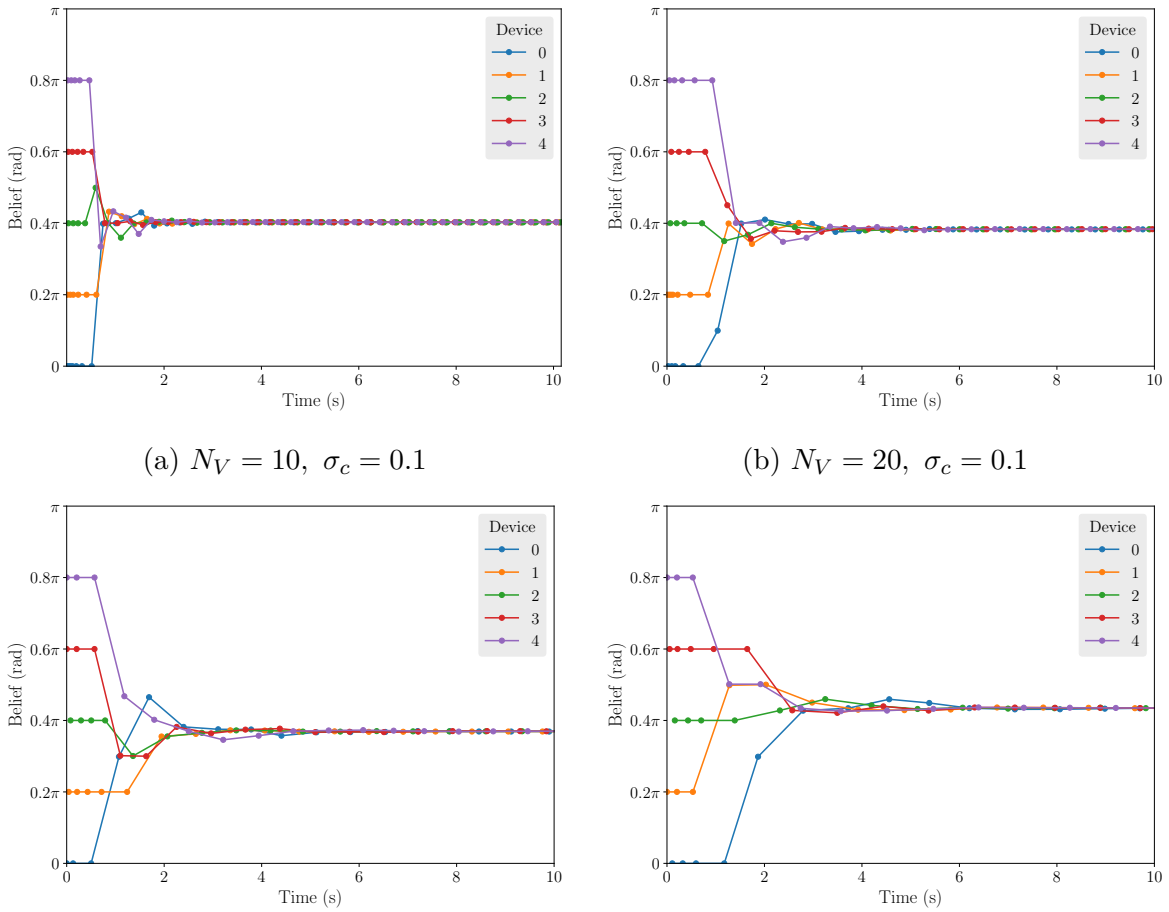
Figure 6.5: Effect of consensus factor strength σ_c on the convergence of $N_R = 5$ devices each with $N_V = 1$ variable with prior factor strength fixed at $\sigma_p = 1.0$. Larger values of σ_c correspond to a *weaker* consensus factor relative to the prior, leading to incomplete convergence towards the group consensus. The experiment duration is capped at $t = 2$ s.

The strength of the prior factor f_p is fixed at $\sigma_p = 1.0$ rad, while the strength of the consensus factor σ_c is varied. A stronger consensus (corresponding to a lower value of σ_c) enforces tighter agreement between connected variables.

As the experiment was executed across separate physical devices, time is aligned such that $t = 0$ s corresponds to the activation of the earliest device. The results, shown in Figure 6.5, are consistent with expectations: when the consensus factor strength is equal to the prior strength ($\sigma_c = \sigma_p$), the belief means converge towards the group average, though

they do not coincide exactly. Small discrepancies arise due to the inherent stochasticity of the setup, as devices activate at slightly different times, operate with independent clocks, and communicate asynchronously over a shared wireless channel.

6.4.2 Multiple Variable Consensus



(a) $N_V = 10,\ \sigma_c = 0.1$

(b) $N_V = 20,\ \sigma_c = 0.1$

(c) $N_V = 30,\ \sigma_c = 0.1$

(d) $N_V = 40,\ \sigma_c = 0.1$

Figure 6.6: Effect of increasing the number of per-device variables N_V on convergence to the group mean. As N_V grows, each device exchanges larger messages, which increases processing time on the ESP32. Consequently, within the fixed experiment duration of $t = 10$ s, devices can complete fewer GBP iterations.

Real-world problems (as discussed in the previous chapters) are likely to involve GBP with multiple variables and factors per device. This naturally increases the volume of

messages (and therefore data) exchanged between devices. To investigate the scalability of the current setup, we vary the number of variables N_V per device while keeping $\sigma_p = 1.0$ and $\sigma_c = 0.1$. We then observe the evolution of the belief means over a fixed duration of $t = 10$ s.

Figure 6.6 shows the belief means of the *first* variable $\mathbf{x}_0^d$ for each device d over time, with $N_V \in \{10, 20, 30, 40\}$. Each data point corresponds to the belief mean at a single GBP iteration. The results clearly indicate that as N_V increases, convergence to consensus takes longer. This slowdown arises because larger amounts of data must be transferred, causing each GBP iteration to take more time. We hypothesise that performance could be significantly improved through more efficient engineering of the ESP32, leveraging its multiple cores and asynchronous programming features (e.g., queues, buffers, and semaphores). In the current implementation, a received message triggers an interrupt, which blocks all other ESP32 processes.

6.4.3 Choice of Floating Point Representation

By default, the `numpy` Python library represents floating-point numbers using 64-bit precision, with each value occupying 8 bytes. This has a substantial impact on the size of messages exchanged between devices. For instance, the size of a single Node2Node message is given by

$$\text{size}\left(\texttt{Node2Node}\right) = \underbrace{1}_{\text{size}(n)} + 2 \times \underbrace{4}_{\text{size}\left(\texttt{node key}\right)} + F\left(\underbrace{n + \frac{n(n+1)}{2} + n}_{\text{size}(\boldsymbol{\eta}+\text{ser}(\boldsymbol{\Lambda})+\boldsymbol{\mu})}\right), \tag{6.9}$$

where n denotes the degrees of freedom of the Lie group and F is the size of one floating point number.

However, 64-bit precision (`float64`) is excessive for a consensus problem on SO(2), par-

ticularly when an acceptable convergence threshold can be set at the order of 0.01 rad. To address this, all experiments in this chapter use 16-bit floating-point representation (`float16`) instead. This dramatically reduces the data size, and for the class of consensus problems considered here, the savings in communication overhead clearly outweigh the marginal loss in numerical precision.

Table 6.1 shows a comparison in using `float16` vs `float64` representation in our message passing, for the same experiment in Section 6.4.2. We record the number of iterations completed in the $t = 10$ s duration, and it is clear to see that using a lower precision of 16-bit floating point numbers increases the computation possible in the time period. This effect is smaller at smaller N_V as the processing time per iteration also incurs some fixed overhead in message packing, chunking, and transmission between devices.

Table 6.1: Effect of floating-point representation on the rate of GBP message passing. As the number of per-device variables N_V increases, the average number of iterations completed in $t = 10$ s decreases. Using 16-bit floating-point representation for inter-device messages allows a higher iteration rate compared to the default 64-bit `numpy` representation.

	Iterations/s	
$\boldsymbol{N_V}$	**16-bit float**	**64-bit float**
1	16.4	12.2
2	12.1	7.7
5	7.3	4.0
10	4.2	2.2
20	2.5	1.5

6.4.4 Asynchronicity

We have shown that one of the main advantages of GBP is that it does not require global synchronisation or a shared clock. Messages can be exchanged in an ad hoc manner, and consensus emerges locally based on the *currently available information*. In this qualitative experiment, we consider $N_R = 3$ devices, each with $N_V = 1$ variable, initialised in the

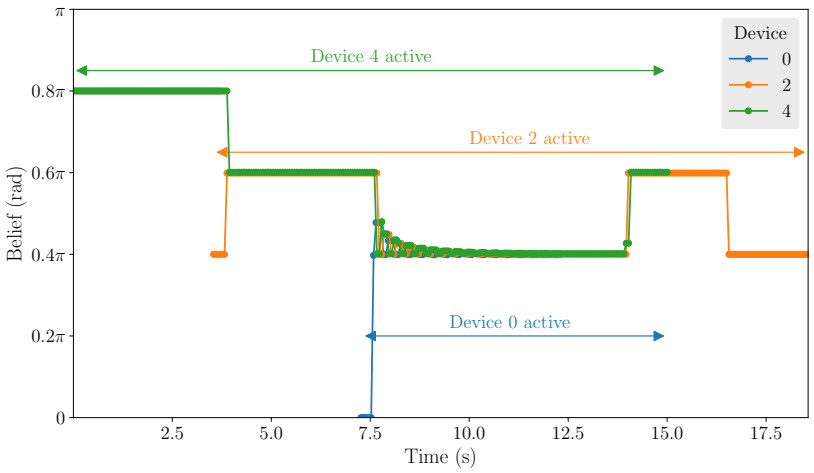


Figure 6.7: Beliefs of the variables of $N_R = 3$ devices over time as the devices activate and deactivate. At any time, consensus is achieved as the average of all active devices. Device 1 is active from $t = 0$ to $t = 14$ s, device 2 is active from $t = 3$ s onwards, and device 4 is only active between $t = 7.3$ to $t = 14$ s.

same manner as in Section 6.4.1. The devices are arbitrarily switched on and off, and their consensus behaviour is observed (Figure 6.7). Initially only device 0 is on with initial belief mean 0.8π rad, and device 2 is switched on at $t = 3$ s with an initial belief mean of 0.4π rad. Immediately the two devices immediately converge to their average value of 0.6π rad. At $t = 7.3$ s device 0 is initialised with belief mean 0 rad, shifting the group consensus to 0.4π rad. This consensus holds until $t = 14$ s, when device 0 is switched off. Finally, at $t = 16.5$ s, device 0 is switched off, leaving only device 2 active. This experiment highlights the asynchronous nature of GBP: consensus is continually updated as devices join or leave the network, with each device performing inference solely on the information available at that time. This property makes GBP particularly well-suited for real-world multi-robot systems, where synchronisation across all agents cannot be guaranteed.

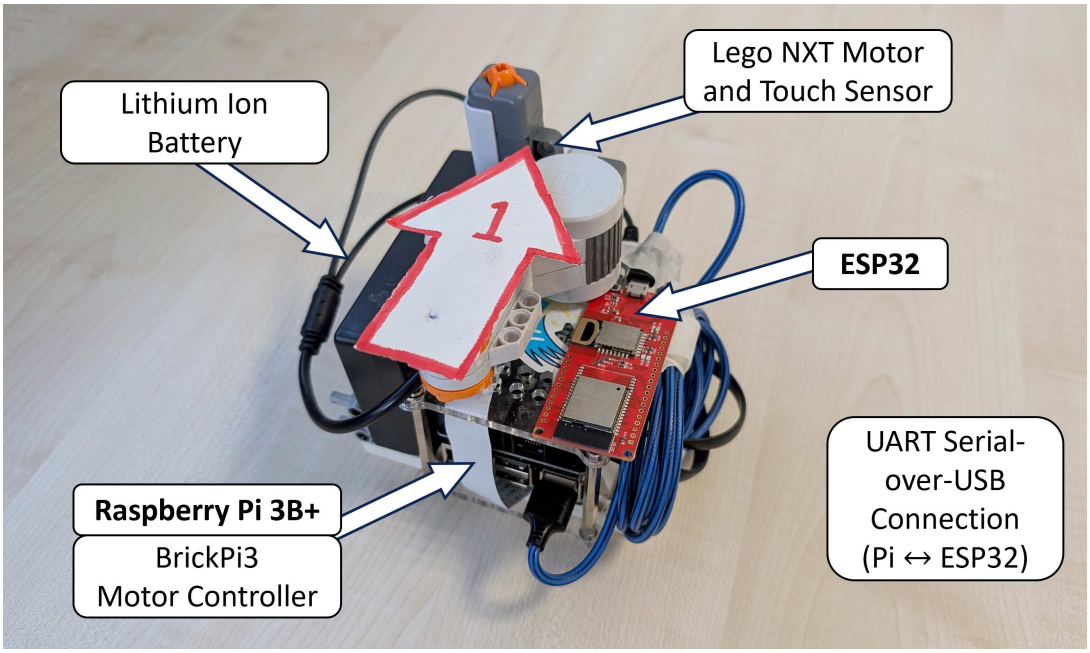


Figure 6.8: The device used in the real-world multi-robot demonstration consists of a Raspberry Pi 3B+ (the processor) and an ESP32 (the microcontroller). The processor is also connected to a motor to visually depict the current belief as a rotation angle, and a touch sensor to enable activating and deactivating the algorithm. A Lithium ion battery is used to power the device.

6.4.5 Demonstration

As a final qualitative experiment, we present a demonstration with five physical devices connected to motors. Each device as shown in Figure 6.8 consists of a Raspberry Pi 3B+ paired with a BrickPi3 motor controller, and an ESP32 microcontroller connected via a UART serial-over-USB interface. Each device is battery powered, ensuring portability. A push-button switch is included to activate or deactivate the GBP algorithm: when *deactivated*, the ESP32 ceases broadcasting polling messages. This avoids the long power-on delay associated with the Raspberry Pi while still allowing devices to rejoin consensus quickly.

As in Section 6.4.1, each device d maintains $N_V = 1$ variable. During the demonstration, the current belief mean of each device's variable is transmitted as a desired angle

command to its motor, providing a visual indication of consensus: when agreement is reached, all devices 'point' in the same direction.

In Figure 6.9 we have placed the 5 devices outdoors. They power on, activate, and achieve consensus without the need for communication over any WiFi network. While intentionally simple, this demonstration serves as a proxy for more complex robotic platforms. Importantly, it highlights the flexibility of the Python-based GBP implementation, which can be readily adapted or integrated into existing robotic hardware and software systems.

Figure 6.9: Multi-robot consensus demonstration in the wild. 5 devices are initialised with prior angles $\mathbf{z}_d \in \mathrm{SO}(2)$ indicated by their arrows (left). When the devices switch on, they create inter-device factors with their neighbours and achieve consensus on a global direction to point towards (right). Using the ESP-NOW protocol removes the need for a central WiFi network for communication and control.

6.5 Discussion

This architecture of using a processor in combination with an ESP32 microcontroller opens new avenues for scalable, infrastructure-free collective inference. By relying on peer-to-peer communication and off-the-shelf protocols such as ESP-NOW, this setup avoids the disadvantages of centralised networking while remaining cost-effective. The

experiments show that even with the packet size limitations of ESP-NOW, Gaussian Belief Propagation (GBP) can be executed on real devices reliably for modestly sized problems. This suggests that decentralised probabilistic inference is feasible on hardware platforms that are both affordable and widely accessible. These results open several directions for future development:

6.5.1 Improving Hardware and Communication Protocols

While ESP-NOW is sufficient for the experiments in this chapter, packet size limitation imposes constraints. Future work could explore microcontrollers with larger memory and faster processing capability, enabling higher-frequency updates over larger factor graphs. At the same time, even with the ESP32, there is significant scope for improvement simply through better embedded software practices. In this thesis, the focus was on making the system work end-to-end rather than fine-tuning low-level code. For instance, the current implementation performs heavy computation inside the ESP32's receive callback function. Although the architecture is not designed for this, it was done here out of expedience and time constraints. A more careful design, separating communication from processing and exploiting the ESP32's concurrency features, could substantially reduce delays and improve robustness. In short, a deeper engineering effort on the existing hardware might already yield major performance gains, without necessarily requiring more powerful devices.

6.5.2 Applications to Distributed Multi-Robot Coordination

A natural direction for future work is to extend this real-world GBP framework to the problems studied earlier in the thesis, such as path planning, cooperative exploration, and shape formation. Applying GBP on real hardware to solve these problems would close the gap between the theoretical formulations developed in simulation and their practical

deployment on decentralised, resource-constrained robots.

6.5.3 Incorporating Multi-Modal Sensing

The focus of this chapter has been on demonstrating the proof of concept of distributed inference. Incorporating onboard sensors such as cameras, IMUs, light sensors, or gas detectors would allow robots to fuse local measurements into the collective estimates in real time. Identifying the limits of communication bandwidth and computational load and determining which types of sensors can realistically be supported are important questions for future work.

6.5.4 Scalability

Extending the system to hundreds or even thousands of robots raises fundamental questions that have not yet been addressed. It is unclear what the dominant bottlenecks will be: network congestion from simultaneous peer-to-peer broadcasts, slower convergence rates as the factor graph grows, or inherent limitations in memory and processing power of the microcontrollers. Small-scale demonstrations, such as those presented in this chapter, suggest that the approach is feasible in principle, but scaling up may expose entirely new challenges that are not visible at small scales. At present, it is not even clear whether such scaling is possible with current off-the-shelf hardware, or whether new devices and protocols will be needed. Careful experimentation on progressively larger setups will therefore be crucial, both to uncover the practical limits of the system and to determine whether the constraints are primarily engineering challenges that can be overcome, or whether the underlying algorithm itself requires modification to remain effective at scale.

Chapter 7

U-ARE-ME: Uncertainty-Aware Rotation Estimation in Manhattan Environments

Cooper: *"CASE, get ready to match our spin with the retro thrusters."*
CASE: *"It's not possible."*
Cooper: *"No. It's necessary."*

— *Interstellar*

Contents

Up to this point, the thesis has focused on distributed optimisation for multi-robot coordination. Yet many robotic applications also depend on accurate perception and estimation, especially of geometric quantities such as camera rotation. This chapter presents U–ARE–ME, an uncertainty-aware rotation estimator that predicts camera rotation together with per-axis uncertainty, enabling robust and temporally consistent estimation from sequences of RGB images.

7.1 Introduction

Accurate estimation of camera rotation from a sequence of monocular images is crucial for many computer vision applications, including visual odometry [He et al., 2020], image stabilisation [Morimoto and Chellappa, 1998], and augmented reality [Tatzgern et al., 2014]. Many solutions have been proposed for a variety of sensor setups. For instance, the recently released Apple Vision Pro operates using visual-inertial odometry, relying on both the cameras and the inertial measurement units (IMUs). However, IMUs are prone to drift, are by design not suitable for non-inertial frames of reference and simply may not be available alongside images.

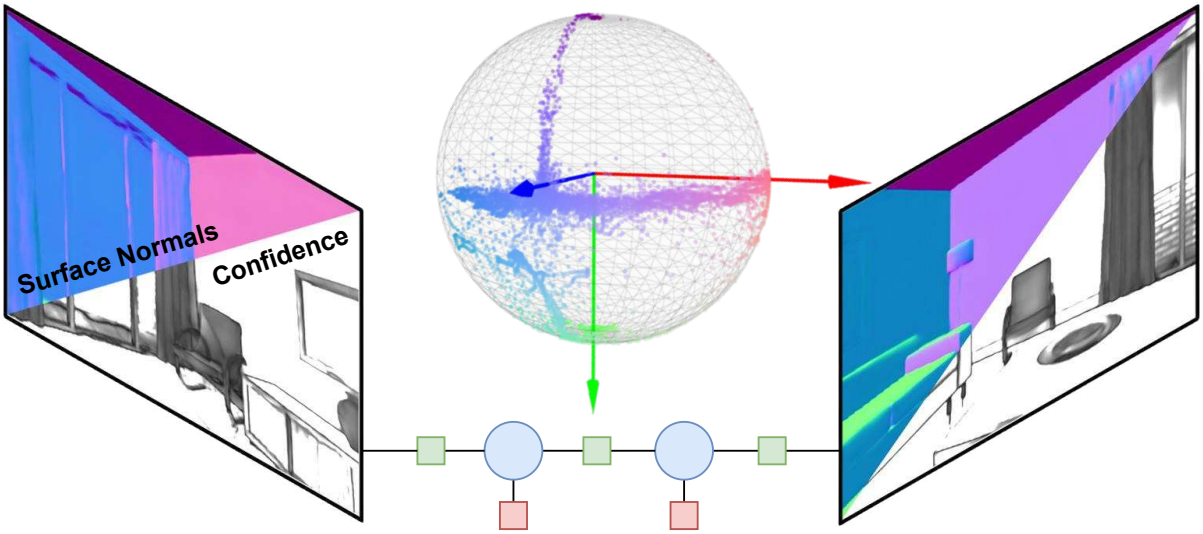


Figure 7.1: U–ARE–ME provides globally consistent rotation estimates in Manhattan environments across sequences of uncalibrated RGB images – no camera intrinsics needed. Rotations are estimated using per-pixel predicted surface normals and confidence. For each frame, we estimate the rotation from the predicted surface normals along with pixel-wise uncertainty and enforce temporal consistency via a factor graph.

If depth measurements (paired with the input images) are available, the RGB-D frames can be aligned — based on photometric and geometric consistency — to recover their relative camera poses [Dai et al., 2017b]. If the surface normal vectors in the scene are aligned with a set of *principal directions*, the camera rotation can be found by aligning the input normals (extracted from the depth maps) to those directions [Silberman et al., 2012, Ghanem et al., 2015b, Straub et al., 2015]. While such approaches provide drift-free rotation estimates with high accuracy, they cannot be applied to in-the-wild videos or devices without a depth sensor.

This chapter focuses on the most challenging setup in which only RGB input is available. Previous attempts have focused on detecting and matching *2D image features*. For instance, ORB-SLAM [Mur-Artal et al., 2015] tracks sparse ORB features, while methods like [Bazin and Pollefeys, 2012, Lee and Yoon, 2015, Pautrat et al., 2023] group line segments to identify the vanishing points (VPs) and hence the camera rotation with respect to the principal directions. However, such methods are sensitive to image degradation (e.g. noise and motion blur) and perform poorly in textureless environments. More im-

portantly, many of these methods assume known camera intrinsics — which are often not available for in-the-wild videos. While a neural network can be trained to regress the rotation between consecutive frames [Cai et al., 2021], such an approach is prone to overfitting and drift. It is also computationally costly to train such a specialised model.

In this work, we propose to make use of the dense pixel-wise geometric priors learned by *single-image surface normal estimation models*. Surface normal estimation models are efficient (e.g. [Bae et al., 2021] runs at ~70+ fps on an NVIDIA 4090 GPU) and have strong generalisation ability [Bae et al., 2021, Bae and Davison, 2024]. In recent years, their usefulness has been demonstrated for various computer vision tasks, including object grasping [Zhai et al., 2023], vision-language reasoning [Liu et al., 2023], simultaneous localisation and mapping [Li et al., 2020b], and CAD model alignment [Langer et al., 2022]. We explore whether such powerful front-end perception can also be used for rotation estimation.

Similar to previous optimisation-based approaches [Silberman et al., 2012, Ghanem et al., 2015b, Straub et al., 2015], we assume a certain distribution of surface normal vectors in world coordinates and optimise for the camera rotation that would align the predicted normals to the principal directions of the scene. While previous methods (1) used depth sensors to extract the normal vectors and (2) were only applicable to a single image, we attempt to remove both constraints.

Two types of uncertainty arise in the process of removing these two commonly adopted constraints. First is the heteroscedastic aleatoric uncertainty [Kendall and Gal, 2017] in surface normal predictions. As shown in [Bae et al., 2021], surface normals predicted by a neural network — unlike those extracted from a depth map — are unreliable, especially for the pixels near object boundaries and on small objects. As these pixels should be down-weighted in the optimisation objective, we introduce a new uncertainty-weighted cost function and show how the uncertainty can be learned in a data-driven manner.

The second type of uncertainty arises when the image contains a limited number of prin-

cipal directions. For instance, when a Manhattan World (MW) [Coughlan and Yuille, 2000] is assumed, two (or more) of the six directions $(\pm X, \pm Y, \pm Z)$ should be observed to determine the camera rotation. If only one axis is visible, any rotation around that axis would result in an equally valid prediction. To this end, we quantify the uncertainty around each principal axis and use it to enhance the temporal consistency in the predictions.

To summarise, our framework alternates between two optimisation steps:

- **Single-frame optimisation:** We optimise the world-to-camera rotation matrix such that the rotated principal directions are best aligned with the predicted surface normals. We improve the accuracy and robustness by introducing an uncertainty-weighted cost function.

- **Multi-frame optimisation:** We take the covariance matrix of rotation around each axis — which is readily available from the Hessian approximation in the second-order optimisation of the first step — and use it to jointly optimise a sliding window of previous frame rotations. We improve the global consistency of our solution, reject outlier rotations and intuitively handle frames that may contain limited information on certain principal axes.

The proposed method runs at ~60+ fps on an NVIDIA 4090 GPU. Note that, unlike the learning-based models that can only be used for rotation estimation, the surface normal predictions — from which we infer the rotation — can be used for other tasks, reducing the overall computational overhead.

The main strength of our approach lies in its *robustness*. Compared to the methods that rely on sparse feature tracking or line segment detection, our approach is more robust to the presence of image degradation. Unlike SLAM-based methods, our approach does

not require camera intrinsics and can be applied to a single image or in-the-wild videos, making it useful in a wider range of scenarios[1].

7.2 Related work

Rotation estimation of a camera from single images has been extensively studied and is generally based upon a 'Manhattan World' (MW) assumption: that indoor scenes exhibit inherent structure, where important features such as walls, floors, ceilings, and edges are aligned with three dominant, orthogonal directions known as the Manhattan Frame (MF). The approaches for (MF) estimation broadly fall into two domains; using RGB images to extract perspective cues, and using 3D information such as surface normals from RGB-D images.

Earlier work estimated the MF by considering vanishing points and lines in an image as perspective cues [Košecká and Zhang, 2002]. The work in [Lee et al., 2009] generated several MF hypotheses from an image, using line segments to find the best fitting model. In [Furukawa et al., 2009], line segments were extracted onto a hemisphere, and clustered to identify three orthogonal directions although this method was sensitive to the chosen resolution of the hemisphere discretisation. The algorithm proposed by [Elloumi et al., 2014] used line clustering to find three vanishing points, and achieved real-time camera rotation estimation over a sequence of images in a video. The work in [Lee and Yoon, 2015] does not rely on the MW assumption but uses sequential Bayesian filtering to jointly estimate rotation and vanishing points. Recently a Hybrid Vanishing Point algorithm (H-VP) [Pautrat et al., 2023] was presented for uncalibrated images, which can extract the MF by making use of a gravity-direction prior to robustly extract vanishing points. These RGB methods rely on the existence of multiple parallel lines and vanishing points, and are not robust in the presence of noise and outliers, or in texture-less scenes.

[1]We encourage the reader to view the demo video on our project page at https://callum-rhodes.github.io/U-ARE-ME/.

Rotation estimation methods that use RGB-D images are more accurate and stable as they utilise 3D information in the scene, whether this is directly through depth camera data, or by using this data to computing the surface normals in the scene. The approach in [Silberman et al., 2012] uses point normals and perspective cues to perform an Exhaustive Search (ES) over a set of candidate directions and using a scoring heuristic to estimate the MF, although this incurs a high computational cost. Depth data from a Kinect camera was used in [Taylor and Cowley, 2012] to determine the MF of a scene by identifying the ground, and selecting a perpendicular direction from one of the walls. This method relies on the presence of a visible floor and multiple walls in the image, and so is not generalisable. In [Ghanem et al., 2015b] the MF is estimated through non-convex optimisation by considering the sparsity constraints of MF-aligned surface normals.

In [Straub et al., 2014], the authors argue that real-world scenes contain a Mixture of Manhattan Frames (MMF), which they simultaneously estimate from surface normals calculated from depth data.

These methods are often sensitive to initial conditions and cannot guarantee global optimality, unlike the family of Branch and Bound (BnB) methods which operate in the rotation search space [Bazin et al., 2012, Hartley and Kahl, 2007, Parra Bustos et al., 2016]. These methods guarantee global optimality, but cannot be considered real-time algorithms. Real-time rotation estimation is enabled in [Joo et al., 2019] using the BnB method by maximising the consensus set of inliers over the search space of rotation. Surface normals are discretised on an equi-rectangular plane to generate the Extended Gaussian Image (EGI) [Horn, 1984], from which the BnB approach is used to estimate the MF.

Recently [Zhang et al., 2022] proposed a novel and efficient cost function of Multiple Normal vectors and Multiple MF Axes (MNMA) for MF estimation. The cost function makes use of the vector dot and cross products between the scene surface normals and the axes of the MF leading to an efficient, accurate and real-time algorithm.

All of the methods considered here have used surface normals calculated from depth data from RGB-D images, rather than directly estimating them from an RGB input. Estimating surface normals has traditionally been computationally intensive, producing unreliable results.

7.3 Method

Given a monocular video, our goal is to estimate the per-frame camera rotation relative to the world coordinates. We begin by assuming that the scene satisfies the MW assumption [Coughlan and Yuille, 2000], which is valid for a wide range of indoor/outdoor scenes. Note that it is straightforward to extend our approach to other world assumptions (e.g. Mixture of Manhattan Worlds [Straub et al., 2014], Atlanta World [Schindler and Dellaert, 2004], and Hong Kong World [Li et al., 2023]), given that the principal directions in the world coordinates are known *a priori*.

Our method is named **U-ARE-ME** (Uncertainty-Aware Rotation Estimation in Manhattan Environments), as it can be used to complement or replace I-M-U sensors. We leverage recent advances in single-image surface normal estimation and propose to infer the camera rotation by aligning the predicted normals to the world assumption. In Sec. 7.3.1 we introduce a new uncertainty-aware optimisation objective and show how the uncertainty can be learned from data. For real-time video applications, it is important to ensure temporal consistency in the predictions. We explain in Sec. 7.3.2 the factor graph formulation required for this.

7.3.1 Single-Image Uncertainty-Aware Rotation Estimation

Suppose that a surface normal vector $\mathbf{n}_i \in \mathrm{S}^2$ corresponding to the i-th pixel is aligned with one of the principal directions. Then, for any Manhattan axis $\mathbf{r} \in \{\pm X, \pm Y, \pm Z\}$,

the angle $\theta = \cos^{-1}(\mathbf{n}_i \cdot \mathbf{r})$ should be $\{0°, 90°, 180°\}$. To this end, Zhang et al. [Zhang et al., 2022] introduced a cost function $E(\mathbf{r}|\mathbf{n}_i) = \sin^2\theta \cos^2\theta$, which is visualised in Fig. 7.2. We modify this cost by multiplying it by some confidence measure κ.

$$E(\mathbf{r}|\mathbf{n}_i, \kappa_i) = \kappa_i \sin^2\theta \cos^2\theta \ . \tag{7.1}$$

To learn κ in a data-driven manner, we pre-train a neural network using the following training loss:

$$\mathcal{L}(\mathbf{n}_i^{gt}|\mathbf{n}_i, \kappa_i) = C(\kappa_i) + \kappa_i \sin^2\theta \cos^2\theta \ , \tag{7.2}$$

where $\mathbf{n}_i^{gt}$ is the ground truth and θ is the angular error of the predicted normal $\mathbf{n}_i$. $C(\kappa)$ should be a monotonically decreasing function of κ to prevent the model from estimating $\kappa_i = 0$ for every pixel. Additionally, the second term should be defined only for $0° \leq \theta < 45°$. Otherwise, the loss could be minimised by *increasing the error*. To satisfy such constraints, we assume that the surface normal probability distribution can be parameterised as follows:

$$p(\mathbf{n}_i^{gt}|\mathbf{n}_i, \kappa_i) = \begin{cases} D(\kappa_i) \exp\left(-\kappa_i \sin^2\theta \cos^2\theta\right) & \theta < \frac{\pi}{4} \\ D(\kappa_i) \exp\left(-\frac{\kappa_i}{4}\right) & \theta \geq \frac{\pi}{4} \end{cases} , \tag{7.3}$$

where $D(\kappa_i) = \exp(C(\kappa_i))$.

Then, $\mathcal{L}(\mathbf{n}_i^{gt}|\mathbf{n}_i, \kappa_i)$ can be interpreted as the negative log-likelihood of the above distribution. $D(\kappa)$ is a monotonically increasing function of κ as the distribution should be normalised. During training, the network is encouraged to increase the value of κ for the pixels with lower error, thereby encoding the confidence in the prediction. In Fig. 7.2, we visualise the proposed distribution for different values of κ. As the analytic form for $D(\kappa)$ could not be found, we obtain the values numerically for $\kappa \in [0, 10^5]$ and fit them

using natural cubic splines. We use a lightweight convolutional encoder-decoder architecture [Alhashim and Wonka, 2018] and use the meta-dataset of training data introduced by DSINE [Bae and Davison, 2024]. Given an RGB image input, DSINE outputs a dense map of surface normals, without any uncertainty information. It also uses per-pixel ray direction as input and thus requires camera intrinsics. We removed this dependency and warped the training images such that the principal point is at the center and the field-of-view (FOV) is 60°. Nonetheless, U–ARE–ME generalises well to images taken with different intrinsics (e.g. the video game sequence in the accompanying video at https://callum-rhodes.github.io/U-ARE-ME/ has a FOV of 86°). DSINE also proposed to improve the piece-wise smoothness and crispness of the prediction by recasting surface normal estimation as iterative rotation estimation. We also remove this iterative process and hence improve the efficiency to give real-time estimates. While this degrades the quality of the surface normal prediction, the rotation estimates from our framework stay robust. U–ARE–ME robustly fuses the per-pixel predictions by weighting them with a confidence κ and is thus robust to mild inaccuracies in the surface normal prediction.

Our network estimates two quantities: the per-pixel surface normal vector $\mathbf{n}$ and the corresponding confidence κ. $\mathbf{n}$ is supervised with the angular loss, and κ with the negative log-likelihood defined Equation 7.2. All other training protocols (e.g. batch size, number of epochs, data augmentation, optimiser, and learning-rate schedulers) are the same as in DSINE. The training only took 9 hours on a single NVIDIA 4090 GPU. After training, κ was capped at 100 to prevent the over-confident predictions from dominating the optimisation.

Given an image with a set of estimated normals and confidence $\mathcal{I} = \{\mathbf{N} = \{\mathbf{n}\}_i, \mathbf{K} = \{\kappa\}_i\}$, the next task is to determine the optimal rotation $\mathbf{R} \in \mathrm{SO}(3)$ that gives the relative rotation of the camera to the MW frame. Since normals are predicted densely, evaluating the cost of a rotation against all normals in the image is prohibitively expensive. To avoid this, a single weighted average cost function can be cheaply calculated for the set of all normals and stays fixed throughout optimisation.

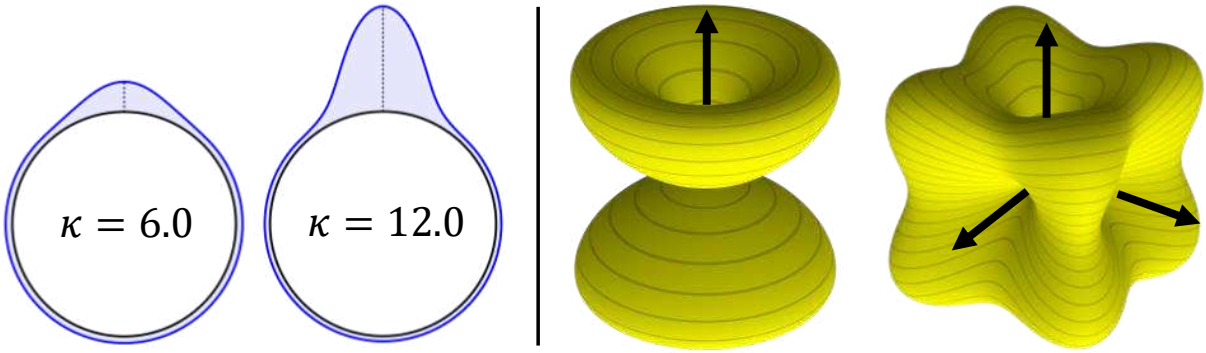


Figure 7.2: **(left)** Visualisation of the proposed surface normal probability distribution for different values of κ. As κ increases, the distribution becomes more concentrated towards the mean direction. **(right)** Visualisation of the cost function defined by a single axis and another one defined by three mutually orthogonal Manhattan axes. The cost is minimised when the predicted normals are parallel or vertical to the axes.

Therefore, using Eq. (7.1), the modified cost function for the optimisation becomes:

$$E(\mathbf{r}|\mathcal{I}) = \frac{1}{|\mathbf{N}| \sum \mathbf{K}} \sum_{i}^{|\mathbf{N}|} E(\mathbf{r}|\mathbf{n}_i, \kappa_i). \tag{7.4}$$

At each stage of optimisation, the cost per X, Y, Z axes of the rotation matrix $\mathbf{R} = [\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z]$ is evaluated to give the total cost:

$$E(\mathbf{R}|\mathcal{I}) = E(\mathbf{r}_\mathrm{x}|\mathcal{I}) + E(\mathbf{r}_\mathrm{y}|\mathcal{I}) + E(\mathbf{r}_\mathrm{z}|\mathcal{I}). \tag{7.5}$$

We use Levenberg-Marquardt (LM) optimisation to minimise Eq. (7.5) (initialised with the identity matrix $\mathbf{I}_3$), rewriting it in terms of the corresponding residual function $f(\mathbf{R})$ (using the parameterisation in [Zhang et al., 2022]) to obtain the optimal rotation $\mathbf{R}^*$.

$$\mathbf{R}^* = \arg\min_{\mathbf{R}} E(\mathbf{R}|\mathcal{I}) \tag{7.6}$$

$$= \arg\min_{\mathbf{R}} f(\mathbf{R})^\intercal f(\mathbf{R}). \tag{7.7}$$

During optimisation, the analytical Jacobian of the residual function, $\boldsymbol{J}_f(\mathbf{R})$, is calculated with respect to a perturbation ξ on the tangent plane (Lie algebra) at the rotation matrix $\mathbf{R}$ i.e. $\mathbf{R} \circ \mathrm{Exp}(\xi)$, where $\mathrm{Exp}(\cdot)$ denotes the exponential map of the SO(3) group. Using the Hessian approximation from the Jacobian in LM optimisation, we can acquire an approximate measure of the covariance of the converged rotation matrix $\Sigma_{\mathbf{R}^*}$ as:

$$J_f(\mathbf{R}^*) = \frac{\partial f(\mathbf{R}^*)}{\partial \xi} \tag{7.8}$$

$$\Sigma_{\mathbf{R}^*} = H^{-1} \approx [\boldsymbol{J}_f(\mathbf{R}^*)^\intercal \boldsymbol{J}_f(\mathbf{R}^*)]^{-1} \tag{7.9}$$

where H is the Hessian matrix approximation.

Having obtained the optimal Manhattan World (MW) rotation $\mathbf{R}_{\mathrm{mw}} = \mathbf{R}^{*\intercal}$ and the uncertainty about this frame, the estimate can then be used for further downstream tasks. This could be for bootstrapping visual odometry systems, correcting drift in inertial pipelines, rectifying images for CNNs, which are sensitive to image rotation, as well as camera calibration to name a few. In the following section we address one such extension, that being to estimate camera rotation across a sequence of images to achieve a coherent trajectory.

7.3.2 Multi-frame Rotation Estimation for Temporal Consistency

When estimating single-frame rotation, our method finds the optimal rotation relative to Identity. The problem with applying this method consecutively to a sequence of images is that there will be no temporal consistency between rotations, and therefore when rotating around any single axis, several MW configurations could satisfy the normal distribution. It is simple to naively initialise the current rotation $\mathbf{R}_t$, with the optimised rotation from the previous frame $\mathbf{R}_{t-1}$ which removes this ambiguity and speeds up convergence of the optimisation. However, two main problems arise from this initialisation. Firstly, for frames that are less Manhattan, we have no way of loosening the MW assumption

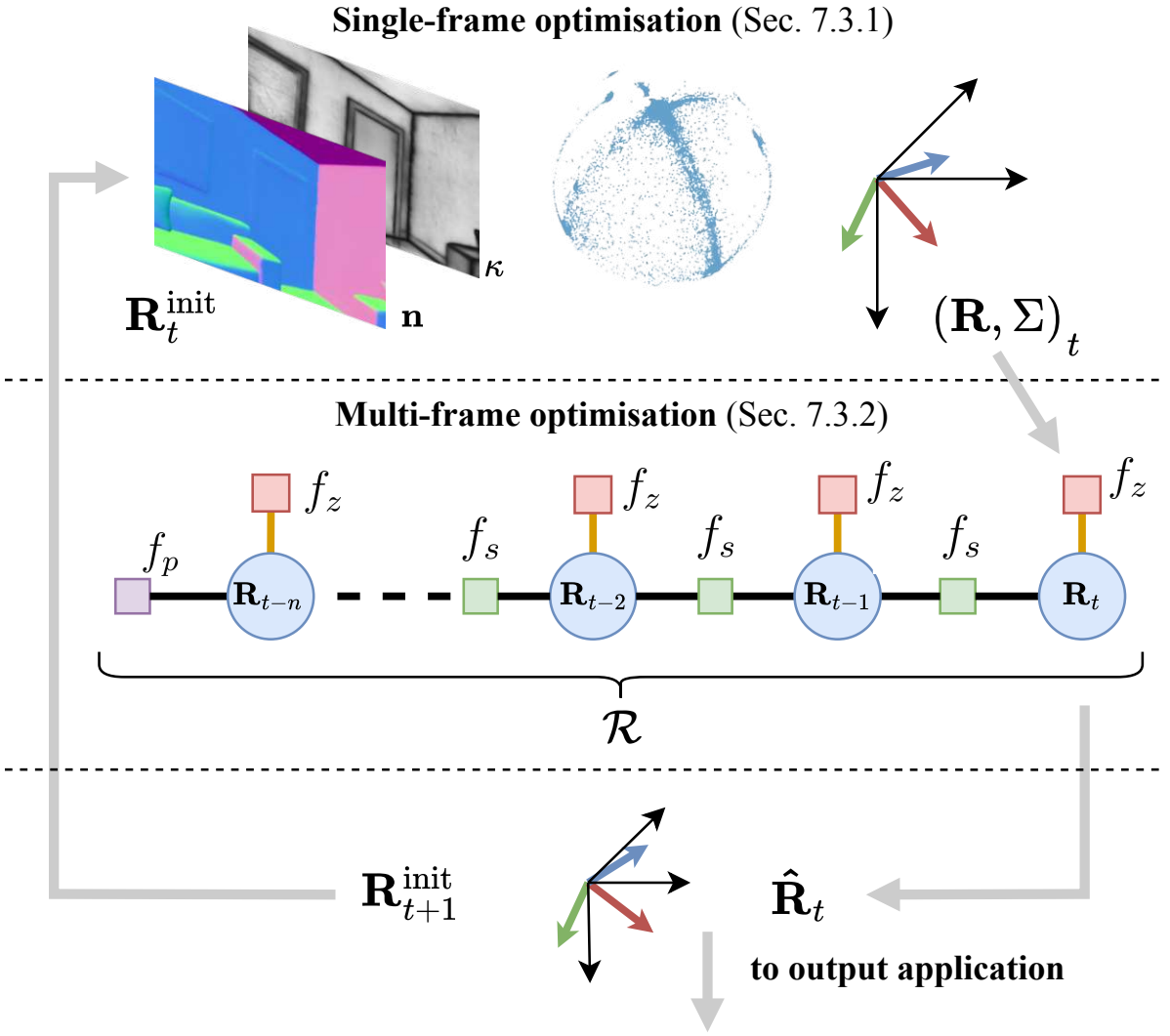


Figure 7.3: The multi-frame optimisation process. Single-frame rotation and covariance estimates are used to initialise a sliding window factor graph in order to provide temporal consistency between frames and reject outlier measurements. Robust factors are shown along orange edges on measurements. The latest frame is then used to initialise the rotation estimate for the next frame.

and therefore a single frame's rotation is only dependent on its own (potentially poorly defined) cost function. Secondly, not all frames in a sequence may optimise to a consistent minimum and may give erroneous rotations that initialise subsequent frames with a bad orientation, potentially poisoning the rest of the sequence. In the following section, we address these issues in order to extend rotation estimation to sequences of images.

7.3.2.1 Sliding Window Optimisation

To tackle both of these issues, we implement a sliding window optimisation that jointly considers previously estimated frame rotations and acts as a support for the current rotation estimate. To model this non-linear joint optimisation, we employ a simple factor graph [Dellaert et al., 2017] that consists of factor nodes $f \in \mathbf{F}$ and variables $\mathbf{X} = \{\mathbf{x}_t, \ldots, \mathbf{x}_{t-n}\}$, where t refers to the latest variable and n defines the length of the sliding window. The joint distribution $p(\mathbf{X})$ is represented as a product of all factors in the graph which, when considering a Gaussian factor graph, can be solved by the following minimisation:

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \sum_{i} ||h_i(\mathbf{X}_i) - \mathbf{z}_i||^2_{\Sigma_i} \tag{7.10}$$

where $\mathbf{X}_i$ represents the clique of variables corresponding to the factor f_i, $h_i(\cdot)$ represents a function that predicts a measurement based on the state of input variables, and $\mathbf{z}_i$ represents some observed measurement.

For our problem, each variable $\mathbf{x}_i \in \mathbf{X}$ represents an SO(3) rotation $\mathbf{R}_i \in \mathcal{R}$, and 3 types of factors exist to constrain the problem. Firstly, $f_z(\mathbf{R}_i)$, is a prior factor on each variable that is set to the rotation estimated from the single-frame estimation problem, $\mathbf{z}_i \in \text{SO(3)}$. The second factor, $f_s(\mathbf{R}_i, \mathbf{R}_j)$, is a smoothness factor which enforces that adjacent frames should have a similar rotation. Finally, another prior factor $f_p(\mathbf{R}_i)$ is added to the oldest frame in the sliding window which represents the marginalised state of the oldest variable in the previous sliding window optimisation, $\mathbf{R}_p$. This multi-frame factor graph is shown pictorially in Fig. 7.3. The minimisation for our problem is therefore given by:

$$\hat{\mathcal{R}} = \arg\min_{\mathcal{R}} \sum_{(i,j)\in\mathbf{F}_s} ||\mathbf{R}_i \ominus \mathbf{R}_j||^2_{\Sigma_{s,i}} + \sum_{i\in\mathbf{F}_z} ||\mathbf{R}_i \ominus \mathbf{z}_i||^2_{\Sigma_{z,i}} + \sum_{i\in\mathbf{F}_p} ||\mathbf{R}_i \ominus \mathbf{R}_p||^2_{\Sigma_{p,i}} \tag{7.11}$$

where $\hat{\mathcal{R}}$ is the optimal set of per-frame rotations, and $\ominus$ denotes the vector increment (defined on the tangent space of the right hand variable) between two rotations via the logarithmic map. See [Sola et al., 2018] for more details on the subject of lie theory when applied to 3D rotations. For ease of implementation, we use the popular sensor fusion library GTSAM [Dellaert and Contributors, 2022] to optimise the multi-frame factor graph based on the definitions above.

Once $\hat{\mathcal{R}}$ has been computed, the latest frame's rotation can then be fed back into the single frame optimisation as initialisation for the subsequent frame i.e. $\mathbf{R}^{\text{init}}_{t+1} = \widehat{\mathbf{R}}_t \in \hat{\mathcal{R}}$

7.3.2.2 Uncertainty and Robust Estimation

To address the issue of non-Manhattan frames, the covariance estimate from Eq. (7.9) can be directly applied to each measurement factor (Σ_z in Eq. (7.11)), since Eq. (7.8) is defined around the global MF upon which we are optimising. Subsequently, frames which exhibit strong Manhattan normals will have a greater influence on the result, stabilising rotation estimates for non-Manhattan frames.

For the smoothness factors, the covariance Σ_s is set to an isotropic covariance $\lambda\mathbf{I}_3$, where λ is a tuning parameter that defines how strongly the smoothness constraint is enforced. The covariance for the prior measurement Σ_p, is automatically defined by the marginalisation of the last variable $\mathbf{R}_{t-n}$ in the previous iterations window.

In order to reduce the influence of outlier measurements due to dropped frames, poor normal predictions, and incorrect local minima from the single frame optimisation process,

we also apply robust factors to all prior measurement factors f_z. We leverage the Huber cost function which represents a Gaussian energy for small residuals, but transitions to a linear function for large residuals. This effectively dampens the influence of measurements that grossly disagree with the smoothness model between variables. However, by maintaining these measurements in the factor graph, a genuine large change in rotation will be properly estimated after a few frames of consistent measurements.

7.4 Experiments

Following previous methods [Li and Tombari, 2022, Kim et al., 2018, Guo et al., 2019], we evaluate the accuracy and robustness of our method on ICL-NUIM [Handa et al., 2014] and TUM RGB-D [Sturm et al., 2012] which cover synthetic and real indoor scenes, respectively. To assess the wider performance in challenging real-world scenarios, we also evaluate the performance on ScanNet [Dai et al., 2017a]. ScanNet images have a significant amount of noise and blur as they were captured using hand-held cameras in poorly lit environments. The scenes are also cluttered with many objects that violate the Manhattan World assumption. We do not discard such scenes and evaluate on all 100 test sequences.

We also demonstrate how our method can be used for up-vector estimation and compare against methods specifically designed for this purpose. In addition, we present results on ground segmentation and horizon estimation, showing the broader applicability of U–ARE–ME.

To measure accuracy, each frame is fed into the algorithm sequentially and the estimated rotation is recorded after each frame. The rotations are then aligned with the relevant ground truth so that methods that do not estimate a specific world alignment can also be compared with the MW-based methods. The metric used for accuracy is the average

rotation error (ARE) and is given by

$$\text{ARE} = \cos^{-1}\left(\frac{\text{tr}(\mathbf{R}_{\text{gt}}^{-1}\hat{\mathbf{R}}) - 1}{2}\right) , \tag{7.12}$$

where $\hat{\mathbf{R}}$ is the rotation estimate and $\mathbf{R}_{\text{gt}}$ is the ground truth.

We compare our approach to various monocular rotation estimation methods. Since we are the first method to directly use learnt normals from monocular images, we compare against other normal optimisation methods designed for RGB-D sensors replacing depth with our predicted normals: RMFE [Ghanem et al., 2015a], RTMF [Straub et al., 2015], ES [Silberman et al., 2012]. This is to show the value of our novel cost function.

We also compare against a recent hybrid vanishing point method (H-VP) for uncalibrated images [Pautrat et al., 2023], which uses a gravity prior to extract more potential VPs (tested with and without the prior). As in the original paper, since H-VP only extracts the Manhattan VPs with no specific X, Y, Z assignment, rotation axes are assigned based on the nearest axis to the ground truth rotation (which we don't rely on). This would not normally be possible on in-the-wild data, giving H-VP perfect temporal consistency across the sequence – thus is comparable to our multi-frame approach.

Furthermore, direct comparisons are drawn with the popular monocular SLAM system ORB-SLAM [Mur-Artal et al., 2015]. Whilst this is not a single image rotation estimation method and requires accurate intrinsics, it is still one of the most widely used methods for obtaining accurate real-time odometry and provides a challenging benchmark from which to draw conclusions of our work.

Lastly, several RGB-D methods (GOME [Joo et al., 2019], Compass [Kim et al., 2018] and E-Graph [Li and Tombari, 2022]) are also used in our comparisons so that we can give context on how accurate our system is compared to methods that require a depth map.

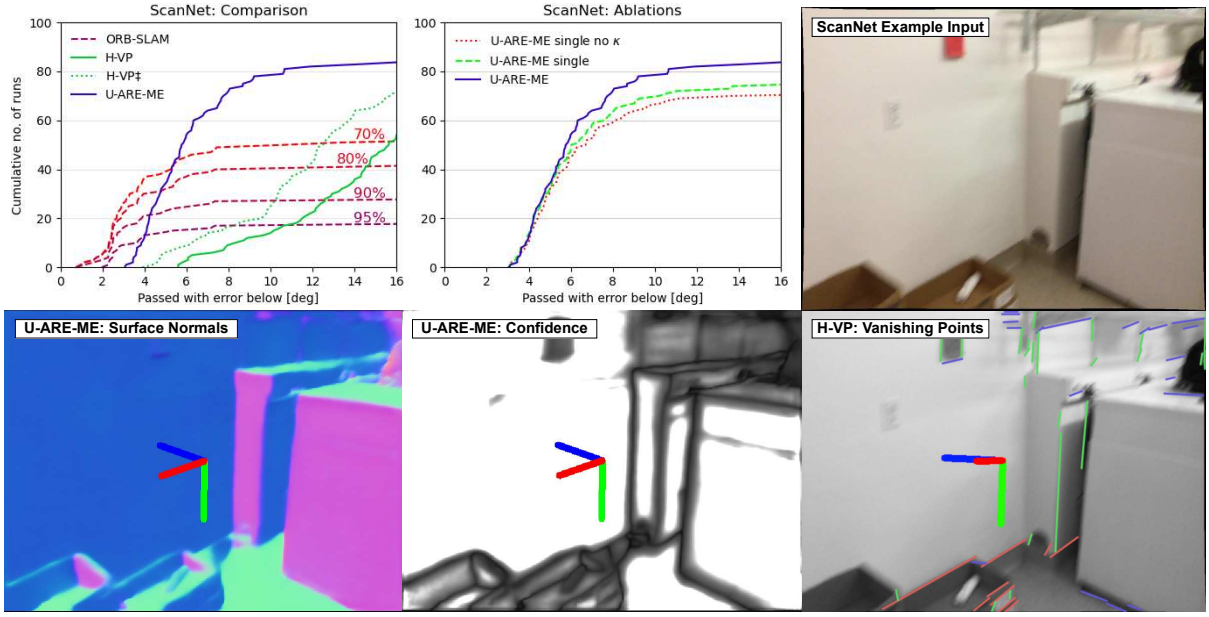


Figure 7.4: **(Top left)** U–ARE–ME accuracy comparison on 100 sequences from the ScanNet dataset. Lines show cumulative number of runs below a certain accuracy threshold. Percentage pass rate is shown for ORB-SLAM, whereby at least X% of frames per sequence must contain a valid rotation estimate (this includes any initialisation and loss of tracking). **(Top middle)** ablation study experiments. The blue line shows the results of the full pipeline. 'single' means that multi-frame optimisation is disabled and 'no κ' means that the uncertainty weighting in the cost function is removed. **(Top right)** example blurred low-texture image from ScanNet, indicative of the challenging scenes contained within. **(Bottom row)** example output from U–ARE–ME and H-VP. Estimated MW rotation from each algorithm are shown centrally, and in the bottom-middle image white indicates a high confidence.

7.4.1 Results from ICL-NUIM and TUM RGB-D

The overall results for both the ICL-NUIM and TUM RGB-D datasets are shown in Table 7.1, comparing our method with popular RGB Single Image methods. U–ARE–ME can accurately estimate a valid trajectory of rotations in 14/15 sequences and has on average the best accuracy of the non-SLAM based monocular methods. Whilst RMFE [Ghanem et al., 2015a], RTMF [Straub et al., 2015] and ES [Silberman et al., 2012] sometimes show the same accuracy as the proposed method, the lack of temporal consistency means that they often shift into a globally inconsistent Manhattan Frame and in many of the

sequences show large errors.

Table 7.1: Quantitative evaluation on ICL-NUIM and TUM RGB-D [deg] for single-image RGB methods. The best method is **bold** and the second-best is underlined. *Italicised* averages don't include failed runs and are not considered for best accuracy metric.

Sequences	**Ours**	**RGB Single Image Methods**				
		RMFE* [Ghanem et al., 2015a]	RTMF* [Straub et al., 2015]	ES* [Silberman et al., 2012]	H-VP [Pautrat et al., 2023]	H-VP† [Pautrat et al., 2023]
Office 0	4.99	4.99	4.97	5.21	1.24	**1.00**
Office 1	3.87	89.18	44.59	3.90	**3.45**	4.79
Office 2	2.38	3.35	2.36	41.99	**0.91**	0.97
Office 3	**2.72**	2.84	41.98	2.87	3.69	3.20
Living 0	8.43	8.36	**8.25**	11.53	×	×
Living 1	**3.58**	91.95	3.81	14.04	7.10	6.37
Living 2	**2.39**	2.45	2.50	2.44	4.17	3.84
Living 3	**5.38**	5.64	5.58	57.62	7.23	6.56
ICL-NUIM Avg.	**4.22**	26.10	14.25	17.45	*3.97*§	*3.82*§
Struc notex	4.61	**4.55**	4.94	7.96	19.10	20.82
Struc tex	**3.03**	3.10	3.18	3.14	11.13	8.25
Large cabinet	4.54	**4.30**	4.60	5.20	7.57	6.82
Cabinet	**5.41**	40.15	6.27	70.39	33.38	20.90
Long office	**5.62**	5.78	5.98	46.74	14.49	12.73
Nostruc notex	**6.93**	54.46	27.52	30.77	×	×
Nostruc tex	28.94	24.16	63.62	**11.58**	31.21	27.81
TUM RGBD Avg.	**8.44**	19.50	16.59	25.11	*19.48*§	*16.22*§

*Reimplemented using our predicted normals †Hybrid VP without the gravity prior §Averages excluding failure sequences

Table 7.2: Quantitative evaluation on ICL-NUIM and TUM RGB-D [deg] for RGB Visual Odometry (VO) and RGB-D methods. *Italicised* averages do not include failed runs and are not considered for the best accuracy metric.

Sequences	**Ours**	**RGB VO**	**RGB-D Methods**		
		ORB [Mur-Artal et al., 2015]	GOME [Joo et al., 2019]	Compass [Kim et al., 2018]	E-Graph [Li and Tombari, 2022]
Office 0	4.99	0.60	5.12	0.37	0.11
Office 1	3.87	×	×	0.37	0.22
Office 2	2.38	0.69	6.67	0.38	0.39
Office 3	2.72	2.53	5.57	0.38	0.24
Living 0	8.43	0.35	×	0.31	0.44
Living 1	3.58	×	8.56	0.38	0.24
Living 2	2.39	0.57	8.15	0.34	0.36
Living 3	5.38	0.84	×	0.35	0.36
ICL-NUIM Avg.	4.22	*0.85*§	*6.81*§	0.36	0.30
Struc notex	4.61	×	4.07	1.96	4.46
Struc tex	3.03	0.37	4.71	2.92	0.60
Large cabinet	4.54	1.13	3.74	2.04	1.45
Cabinet	5.41	×	2.59	2.48	2.47
Long office	5.62	7.86	×	1.75	-
Nostruc notex	6.93	×	×	×	-
Nostruc tex	28.94	17.42	×	×	-
TUM RGBD Avg.	8.44	*6.70*§	*3.78*§	*2.12*§	2.25

§Averages excluding failure sequences

In some sequences the presence of strong line features mean that VP methods retrieve very accurate results (e.g. office scenes due to ceiling tiles), however in less textured scenes and with lower quality images, the line segment detectors of VP methods struggle to acquire enough features to perform RANSAC reliably. As we use a dense normal predictor we don't suffer in these texture-less scenes.

Table 7.2 shows the results comparing U–ARE–ME with RGB visual odometry (RGB-VO) methods like ORB-SLAM, and with methods that use RGB Depth (RGB-D) information.

Comparing to the RGB-D methods, the proposed method is often better than GOME [Joo et al., 2019] despite only using RGB images, and whilst we are worse than Compass [Kim et al., 2018] and E-Graph [Li and Tombari, 2022] for the synthetic ICL-NUIM sequences (which have perfect depth), we achieve comparable accuracy in some real-world TUM sequences.

Comparing to ORB-SLAM, for the ICL-NUIM sequences in which ORB-SLAM does not fail, we find that ORB-SLAM is significantly better than all the RGB methods and is even on par with the RGB-D methods. However, ORB-SLAM may be unable to initialise or lose tracking in texture-less scenes. Such sparse feature-based approaches also suffer from image degradation. To provide a further analysis on how different approaches would perform in challenging, real-world scenarios, we provide the results on ScanNet in the following section.

7.4.2 Results from ScanNet

The ScanNet suite provides a large set of real-world sequences from which we can draw better conclusions about the generalisability of our system. We therefore further test U–ARE–ME on all 100 test sequences and compare again to H-VP and ORB-SLAM. We then also perform an ablation study on the proposed method.

Fig. 7.4 (top left) shows the cumulative number of runs where the mean angular accuracy for a particular sequence is below a threshold. Since ORB-SLAM is a multi-view system and therefore will never produce rotation estimates for every single frame, we show the results of ORB-SLAM at different success rates e.g. 70% defines that at least 70% of frames per sequence need valid rotation estimates (the missing frames being from either initialisation or loss of tracking). In this regard, we allow ORB-SLAM to lose tracking and do not consider this an outright failure.

The results show that U–ARE–ME has a much higher robustness to the ScanNet sequences and will estimate rotations with an accuracy $< 10°$ in 80% of the sequences. Comparing this to ORB-SLAM with a 95% frame threshold which only successfully achieves $< 10°$ in 17% of the sequences. For higher accuracy $< 3°$, we find that ORB-SLAM is more capable in some sequences (10%) whereas the proposed method achieves at best 3°. We argue that this is primarily caused by the accuracy of the surface normal predictions, e.g. the state of the art [Bae and Davison, 2024] reports a mean normal error of 16.2° on ScanNet, and therefore as normal predictions improve so should our results.

H-VP similarly struggles on ScanNet and is actually hindered by the gravity prior – most likely due to the wide range of rotations that violate the gravity assumption. Most of the ScanNet sequences are indoor scenes containing many blank walls which do not provide strong line features or point features. VP methods struggle in these scenes, as shown in Fig. 7.4 where shadow lines are mistakingly classed as MW vanishing points. Our normal network however predicts these regions as planar and also predicts a low confidence (shown in black) so robustly ignores these regions during optimisation. More generally, in low-texture scenes normal prediction networks can rely on subtle lighting cues at the intersection of walls to determine the orientation of these large planar surfaces.

The ablation study Fig. 7.4 (top middle), shows that the overall reliability of the system improves as firstly, the uncertainty-weighted normals reject non-Manhattan pixels within the image, and more so secondly, the multi-frame optimisation provides a consistent

global MF which anchors the solution across the sequence.

7.4.3 Robustness to Out-of-Distribution Camera Intrinsics

To show that our algorithm is robust to a wide variety of out-of-distribution images, a brief study is performed with varying camera intrinsics on the same sequence. We re-evaluate on the ICL-NUIM living room 2 sequence but change the focal length and principal point of the input images, as shown in Fig. 7.5. By performing an increasingly aggressive central crop and then resizing back to the original resolution, a narrowing of the FOV is achieved while keeping a constant principal point. In the shift tests, both the focal length and the principal point vary by cropping from the bottom right corner and resizing.

Table 7.3: Results from modified intrinsics tests.

Crop	Ours	H-VP	ORB-SLAM
Original	2.39	4.17	**0.57**
5% crop	**2.29**	4.57	2.64
10% crop	**2.26**	5.80	4.89
15% crop	**2.26**	6.58	5.64
20% crop	**2.34**	7.40	8.59
25% crop	**2.62**	8.50	13.34
Shift	Ours	H-VP	ORB-SLAM
original	2.39	4.17	**0.57**
5% shift	**2.12**	6.17	4.35
10% shift	**2.50**	7.51	10.27
15% shift	**3.53**	9.25	17.01
20% shift	**4.99**	10.65	26.73
25% shift	**6.41**	12.17	×

Table 7.3 shows the accuracy comparison between U–ARE–ME, H-VP and ORB-SLAM (intrinsics given to ORB-SLAM are kept constant throughout). Since our normal network does not need specific camera intrinsics, we are extremely robust to both image cropping and image shifting – more so for the former. Due to the network being trained on ~60^o

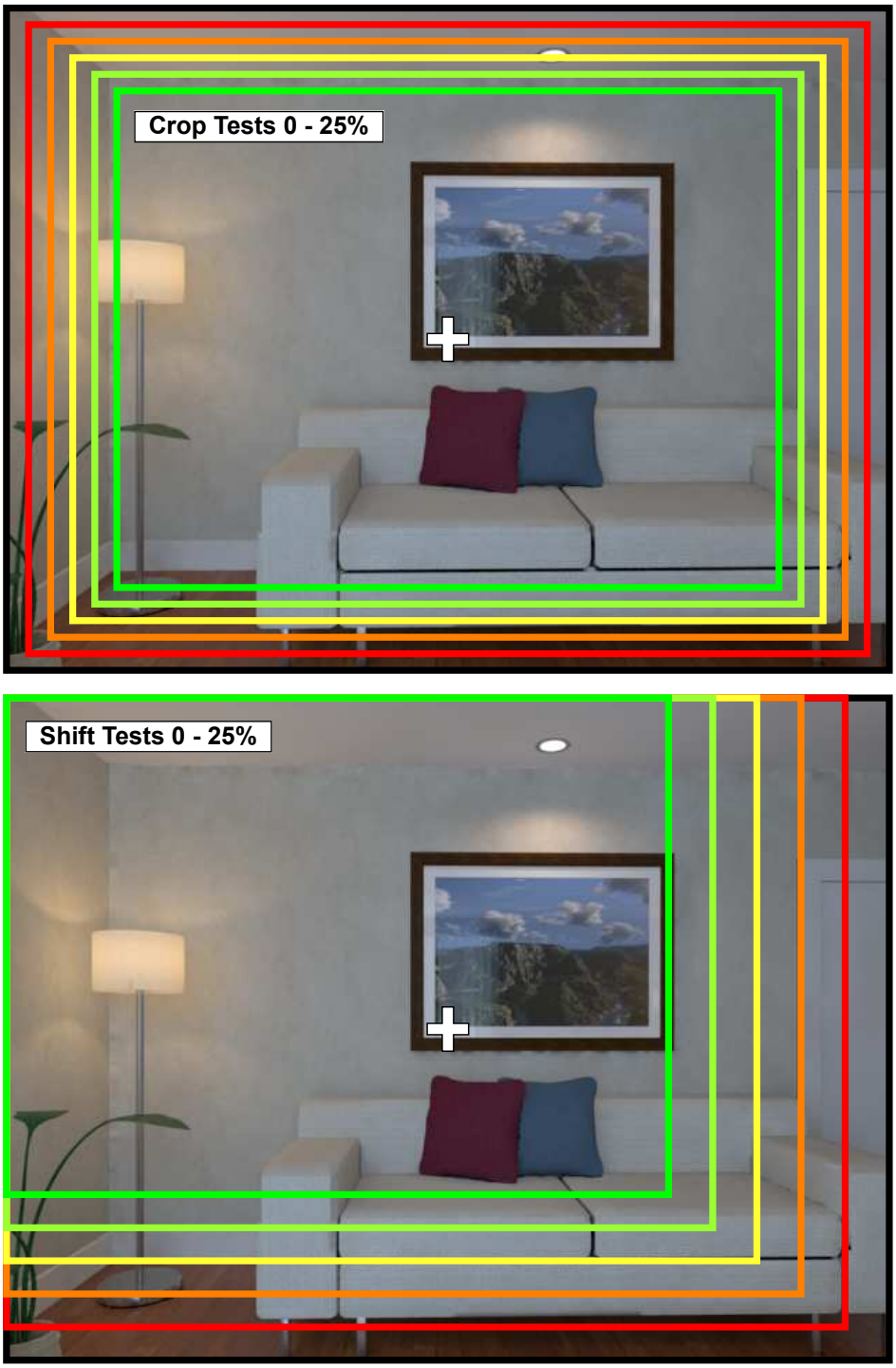


Figure 7.5: Robustness to out-of-distribution images. Crop tests simulate a narrowing focal length with a constant principal point and shift tests simulate both focal length and principal point change. Images are resized to their original resolution after cropping.

Figure 7.6: IMU sensors are prone to drift especially in non-inertial frames of reference (e.g. inside a moving vehicle). The horizon line in each image represents the *up-vector* inferred from our proposed method (middle) and an IMU sensor (right).

FOV images, a minor improvement is observed at ~10-15% cropping as the ICL-NUIM images have a slightly wider than ideal 67^o FOV. As expected, ORB-SLAM rapidly deteriorates with incorrect intrinsics. Despite H-VP not requiring known intrinsics, a narrower FOV in general reduces the number of sparse line features and therefore accuracy degrades proportionally (a major upside to dense predictors).

7.5 Applications

U–ARE–ME can be used on in-the-wild videos without the need for camera intrinsics, even with a weak or non-existent Manhattan World environment. We present a selection of applications of our work, namely up-vector estimation, partial state estimation while operating in a non-inertial frame of reference, as well as ground segmentation. We also provide a video demonstration of U–ARE–ME.

7.5.1 Up-vector Estimation

Estimating the 'upward' direction of a given image is a task that has been tackled by many recent works and can naturally also be extracted from our method by simply dropping the

Table 7.4: Accuracy of the estimated up-vector [°]. Note that PF [Jin et al., 2023] and CTRL-C [Lee et al., 2021] were trained specifically to estimate the up-vector.

Sequence	U–ARE–ME	PF	CTRL-C
Living 0	**7.53**	9.90	12.14
Living 1	**2.67**	3.17	13.74
Living 2	**1.77**	4.67	9.71
Living 3	**4.01**	10.84	7.67
Office 0	**3.95**	5.52	18.87
Office 1	**3.50**	4.65	22.32
Office 2	**1.61**	3.80	15.95
Office 3	**2.13**	3.95	18.47

yaw component of the rotation matrix. Recent neural network approaches have shown success in estimating camera parameters such as the up-vector. CTRL-C [Lee et al., 2021] proposes an end-to-end transformer approach to combine detected vanishing points with learned features. Perspective Fields (PF) [Jin et al., 2023] predicts the per-pixel information about the camera parameters, and demonstrated the use of the up-vector for AR effects such as compositing rainfall and 3D objects into the scene.

We compare our method against these baselines on the ICL-NUIM dataset and report the angular difference of the up-vector in Table 7.4. As these baseline methods only operate on single RGB images, we perform single frame rotation estimation in our method for a fair comparison. Our method outperforms the baselines which were trained on feature-rich image datasets, and is able to more accurately estimate camera rotation in the relatively texture-less scenes from the ICL-NUIM dataset.

7.5.2 Horizon Estimation in Non-inertial Frames of Reference

Motivated by the limitations of using Inertial Measurement Units (IMUs) when operating within a non-inertial reference frame, we apply U–ARE–ME to a real-world video sequence captured from a bus with longitudinally and laterally accelerating motions. We perform multi-frame rotation estimation for the sequence of RGB video frames, and per-

form a comparison against IMU data. For each frame, we compute the ARE between the estimated rotation and the initial rotation, and plot the evolution over time in Fig. 7.6.

The camera is stationary with respect to the bus throughout the sequence so there should be no relative rotation. The Intel Realsense D435i camera was used to take synchronised RGB and IMU measurements which were integrated using the complementary filter provided by the Intel Realsense SDK [Intel, 2024].

As the bus turns harshly, our visual-only approach maintains a steady rotation estimate, while the IMU suffers from strong accelerations causing a large error in its measured rotation. This is seen in Fig. 7.6 where the green horizon lines for each method have been calculated using the up-vector derived from the estimated rotation. We envisage that U–ARE–ME can be used to complement traditional IMU-based applications in non-inertial reference frames.

7.5.3 Ground Segmentation

The up-vector can be used to perform real-time ground segmentation from RGB images, which is an important pre-processing step in many applications including robotics [Man et al., 2018], autonomous driving [Deng et al., 2024], and 3D object tracking for augmented reality [Tatzgern et al., 2014], where it can be used to seamlessly integrate virtual objects with the real-world environment.

As U–ARE–ME produces per-pixel surface normal estimates, we can directly use the result of the rotation estimation to segment areas of an input image corresponding to the ground, assuming that this is aligned with the world up-vector. Our method can be applied to real-world indoor and outdoor images to segment the ground even when the scene is non-Manhattan, as seen in Fig. 7.7.

(a) London (b) Mumbai

(c) Cardiff (d) Office

Figure 7.7: Even in non-Manhattan scenes, our framework can optimise the rotation such that the global up direction is aligned with the surface normal of the ground plane. It can thus be used to accurately segment the ground plane, shown here in green, which can be useful for many applications in robotics.

7.5.4 Demo video

For a visual overview of our method, we encourage the reader to view the video on our project page[2].

For the reader's convenience, the timestamps of each section in the video are summarised in Table 7.5. We provide additional detailed explanations for each section below.

[2]Project Page: https://callum-rhodes.github.io/U-ARE-ME

Table 7.5: Timestamps for the contents of the demo video

Section		**Timestamp**
U–ARE–ME Demo	ICL-NUIM	0:13
	TUM-RGBD	0:30
	Tokyo walking sequence	0:47
	Video game sequence	1:05
Robustness to Bad Frames		1:29
Applications	Non-inertial Reference Frame	1:58
	Ground Segmentation	2:27

7.5.4.1 U–ARE–ME Demo

We first demonstrate the operation of U–ARE–ME on the ICL-NUIM and TUM-RGBD datasets discussed in our paper. The coordinate frame in the center of the video depicts the orientation of the global Manhattan frame. Throughout the demonstration, the video cycles through various visualisations of the scene:

- RGB image input to U–ARE–ME
- Predicted surface normals (using X–Y–Z to R–G–B colour mapping)
- Confidence of predicted surface normals (greyscale – white represents high confidence)

ICL-NUIM: living-room-2. This synthetic scene shows a camera moving through a living room which is generally Manhattan in structure, but does contain some features which do not agree with a Manhattan assumption (e.g. curtains, lamps, and sofa cushions). Of note is the textured wall painting at 0:13 which is predicted as having the same surface normal as the wall it is on, while the painting's frame is predicted to have high uncertainty normals. The curtains seen at 0:21 (a large non-Manhattan area) are also shown with high uncertainty normals as they have irregular geometry. As U–ARE–ME is uncertainty-aware, it estimates rotation with a greater weighting on those surfaces that

agree with the Manhattan assumption, e.g., the walls and floor (shown in white on the confidence images), whilst down-weighting the non-Manhattan features.

TUM-RGBD: fr-3 large cabinet. This dataset contains videos captured with a real-world hand-held camera, exhibiting some pitch and roll with unsteady camera motion. The normals of objects in the background with fine structures and lots of occlusion are harder to accurately predict, and thus are estimated as having surface normals with higher uncertainty. As a result, our method can produce accurate estimates of rotation throughout the sequence by down-weighting such uncertain normals.

Tokyo Walking Sequence. We apply our method to an *in-the-wild* video taken directly from YouTube [YouTube, 2019] showing a hand-held camera viewpoint walking through the streets of Tokyo. This is a challenging situation for rotation estimation as the camera intrinsics are not known. The environment is dynamic, with many pedestrians walking in the scene, and our method produces reliable rotation estimation despite the small quantity of static Manhattan-aligned objects and buildings.

Video Game sequence. In this example, our method is shown to estimate rotation on a synthetic sequence from the video game Star Citizen [YouTube, 2021]. Once again the camera intrinsics are not known, and this is a relatively non-Manhattan environment. During the sequence, the camera switches between 1st person and 3rd person (during which the player character takes up a significant portion of the screen) yet rotations remain consistent with the game world. U–ARE–ME takes this into consideration by estimating a high uncertainty on the player model.

7.5.4.2 Robustness to Dropped Frames

We compare U–ARE–ME operating **with** (bottom videos) and **without** (top videos) multi-frame optimisation, when black frames are randomly injected into the sequence, simulating dropped frames.

For example, at 01:50 the addition of a black frame cause the non-multi-frame rotation estimate (top) to differ significantly from its previous estimate. Using our proposed uncertainty-aware multi-frame optimisation however, U–ARE–ME is seen to produce temporally consistent rotation estimates, in the presence of dropped frames.

7.5.4.3 Applications

Finally, we demonstrate some applications of U–ARE–ME as discussed in the paper.

Operation in a Non-inertial Reference Frame. This section of the demo video shows the benefit of applying U–ARE–ME to a real-world video sequence captured in a non-inertial reference frame for estimating a horizon as discussed in section 7.5.2. The video shows an outward view demonstrating the motion of the bus (left), and the horizon lines estimated by U–ARE–ME (middle) and based on IMU measurements (right).

Ground Segmentation. Further to section 7.5.3, we apply U–ARE–ME to a video taken directly from YouTube [YouTube, 2018]. Without knowledge of the video's camera intrinsics, and in the presence of blurring noise, our method is able to estimate the up-vector and highlight the ground-aligned pixels in green.

7.6 Conclusion

Motivated by the need to provide extrinsic rotation estimates from *in-the-wild* images and sequences, we have presented U–ARE–ME, an accurate and robust camera rotation estimator that operates on uncalibrated RGB images. The system is capable of outputting estimates for single images and has also been extended to reliably handle multi-frame scenarios. An extensive evaluation has been performed and it has been shown that our method is capable of providing globally consistent multi-frame rotation estimates which rivals the performance of similar methods that leverage accurate depth maps – all whilst remaining real-time. By accounting for the uncertainty and inherent ambiguity of the common MW assumption, we are also capable of providing accurate results on scenes that superficially appear to not contain structural regularities, and where other 2D feature-based methods often fail.

Chapter 8

Conclusion and Future Directions

If you think you know it well, then little indeed you know.

— Kena Upanishad 2.3

This thesis has investigated algorithms for distributed collaboration and decision making in multi-robot systems, using Gaussian Belief Propagation (GBP) for decentralised inference over factor graphs. The work is motivated by the increasing computational capabilities and autonomy of individual robots, and the need for coordination frameworks that can operate with ad-hoc communication schedules and avoid centralised control. A core principle behind this work is that each robot should be capable of acting independently, while also coordinating efficiently with its neighbours to enable collective reasoning.

In Chapter 3 we introduced the GBP Planner, a distributed method for multi-robot path planning based on GBP. This demonstrated how coordinated path planning could be achieved through pure peer-to-peer local message passing, enabling efficient, scalable, and safe robot trajectories, even under scenarios of communication failures.

In Chapter 4, GBPStack extended this concept to more complex optimisation problems involving multiple competencies that robots might wish to optimise over. Robots were

able to jointly infer and form a consensus over a global shared state, whilst also optimising over the path planning and exploration sub-problems. This chapter made use of the insight that the sparsity of complex inter-dependent optimisation problems can be exploited by representing each sub-problem as a layer in a factor graph stack.

Chapter 5 further explored multi-robot consensus through DANCeRS, which unified decision-making over continuous and discrete spaces with one message-passing framework. The algorithm demonstrated flexibility over tasks such as path planning for shape formation, and discrete decision-making.

Beyond algorithmic development, this thesis also presented a physical implementation of GBP on low-cost embedded hardware using a peer-to-peer communication protocol. The practical validation in Chapter 6 indicates that high-level coordination could be achieved on resource-constrained platforms without relying on centralised infrastructure, reinforcing the broader vision of truly decentralised autonomy.

Finally in Chapter 7 we presented U–ARE–ME, an uncertainty-aware rotation estimator for uncalibrated sequences of RGB images. While not a multi-robot method, it demonstrated the effectiveness of using factor graphs for perception under uncertainty.

Across the contributions of this thesis, we showed that GBP as an inference tool for factor graph problems can serve as a lightweight yet powerful backbone for multi-robot collaboration, unifying perception and coordination under a distributed, scalable framework.

Looking forward, several important avenues remain open. Having demonstrated GBP on embedded systems, the next step is deploying GBP-based path planning, shape formation, and consensus on small physical robot teams. Yet questions of scalability remain: how many robots can the ESP-NOW solution support before communication bottlenecks hinder optimisation, and what trade-offs exist between graph connectivity, system performance, GBP convergence, and message-passing schedules?

Another promising direction involves extending GBP to dynamic environments, where

the belief structure or the environment itself changes over time. This would be useful in environmental mapping tasks, and could require relaxation of beliefs (through dynamically changing covariances in the factor graph), or through copies of optimisation variables created at each time step, which was lightly touched on in Chapter 5.

As robots become more autonomous, numerous, and capable, the need for scalable and communication-efficient frameworks for joint inference and control will become essential. The methods developed here suggest that factor graphs and distributed message passing can form the backbone of such systems, allowing robots not just to act individually, but to collaborate, negotiate and optimise together.

Bibliography

[Afdila et al., 2023] Afdila, R., Fahmi, F., and Sani, A. (2023). Distributed formation control for groups of mobile robots using consensus algorithm. *Bulletin of Electrical Engineering and Informatics*, 12:2095–2104. 121

[Afghah et al., 2019] Afghah, F., Razi, A., Chakareski, J., and Ashdown, J. D. (2019). Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles. *CoRR*, abs/1905.00492. 23

[Alhafnawi et al., 2021] Alhafnawi, M., Hauert, S., and O'Dowd, P. (2021). Self-organised saliency detection and representation in robot swarms. *IEEE Robotics and Automation Letters*, 6(2):1487–1494. 121

[Alhashim and Wonka, 2018] Alhashim, I. and Wonka, P. (2018). High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*. 178

[Alonso-Mora et al., 2018a] Alonso-Mora, J., Beardsley, P., and Siegwart, R. (2018a). Cooperative collision avoidance for noholonomic robots. *IEEE Transactions on Robotics (T-RO)*, 2(2):404–420. 78

[Alonso-Mora et al., 2018b] Alonso-Mora, J., Montijano, E., Nägeli, T., Hilliges, O., Schwager, M., and Rus, D. (2018b). Distributed multi-robot formation control in dynamic environments. *Autonomous Robots*, 43(5):1079–1100. 24

[apnews.com, 2023] apnews.com (2023). A swarm of small drones may help artificial reefs attract sea life. https://apnews.com/article/cyprus-artificial-reefs-autonomous-underwater-vehicles-uav-0e4a2f73b4cd94dcb3296614500e57e9. 23

[ASTRIIS, 2023] ASTRIIS (2023). ASTRIIS develops a multi-robot adaptive sensing system for ocean observation. https://astriis.pt/astriis-develops-a-multi-robot-adaptive-sensing-system-for-ocean-observation/?utm_source=chatgpt.com. 23

[Atanasov et al., 2014] Atanasov, N., Le Ny, J., Daniilidis, K., and Pappas, G. J. (2014). Information acquisition with sensing robots: Algorithms and error bounds. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6447–6454. 100

[Bae et al., 2021] Bae, G., Budvytis, I., and Cipolla, R. (2021). Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 172

[Bae and Davison, 2024] Bae, G. and Davison, A. J. (2024). Rethinking inductive biases for surface normal estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 172, 178, 191

[Balasubramanian et al., 2015] Balasubramanian, S., Melendez-Calderon, A., Roby-Brami, A., and Burdet, E. (2015). On the analysis of movement smoothness. *Journal of NeuroEngineering and Rehabilitation*, 12(112). 88

[Bazin and Pollefeys, 2012] Bazin, J.-C. and Pollefeys, M. (2012). 3-line ransac for orthogonal vanishing point detection. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4282–4287. IEEE. 171

[Bazin et al., 2012] Bazin, J.-C., Seo, Y., Demonceaux, C., Vasseur, P., Ikeuchi, K., Kweon, I., and Pollefeys, M. (2012). Globally optimal line clustering and vanishing point estimation in manhattan world. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 638–645. 175

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc. 52

[Bizyaeva et al., 2023] Bizyaeva, A., Franci, A., and Leonard, N. E. (2023). Nonlinear opinion dynamics with tunable sensitivity. *IEEE Transactions on Automatic Control*, 68(3):1415–1430. 24, 121

[Brambilla et al., 2013] Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7:1–41. 118

[businessinsider.com, 2025] businessinsider.com (2025). Amazon's sprawling warehouse robot factories offer a glimpse into modern US manufacturing. https://www.businessinsider.com/amazon-robotics-facilities-show-modern-us-manufacturing-2025-5. 23

[Cai et al., 2021] Cai, R., Hariharan, B., Snavely, N., and Averbuch-Elor, H. (2021). Extreme rotation estimation using dense correlation volumes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14566–14575. 172

[Coughlan and Yuille, 2000] Coughlan, J. and Yuille, A. L. (2000). The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *Neural Information Processing Systems (NeurIPS)*. 173, 176

[Dai et al., 2017a] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017a). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 184

[Dai et al., 2017b] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017b). Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1. 171

[Davison and Ortiz, 2019] Davison, A. J. and Ortiz, J. (2019). Futuremapping 2: Gaussian belief propagation for spatial ai. *arXiv preprint arXiv:1910.14139*. 54, 56, 72

[Dellaert, 2021] Dellaert, F. (2021). Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:141–166. 55, 79

[Dellaert and Contributors, 2022] Dellaert, F. and Contributors, G. (2022). GTSAM. https://github.com/borglab/gtsam. 183

[Dellaert et al., 2017] Dellaert, F., Kaess, M., et al. (2017). Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139. 182

[Deng et al., 2024] Deng, W., Chen, X., and Jiang, J. (2024). A staged real-time ground segmentation algorithm of 3d lidar point cloud. *Electronics*, 13:841. 196

[Deray and Solà, 2020] Deray, J. and Solà, J. (2020). Manif: A micro lie theory library for state estimation in robotics applications. *Journal of Open Source Software*, 5:1371. 64, 67, 124, 131

[Dias et al., 2006] Dias, M., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270. 24

[Drummond and Cipolla, 2000] Drummond, T. and Cipolla, R. (2000). Application of lie algebras to visual servoing. *Int. J. Comput. Vis.*, 37(1):21–41. 64

[Duisterhof et al., 2021] Duisterhof, B. P., Li, S., Burgues, J., Reddi, V., and de Croon, G. (2021). Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In *Proceedings of the IEEE Conference on Intelligent Robots and Systems (IROS)*. 101, 146

[Ebert et al., 2022] Ebert, J. T., Berlinger, F., Haghighat, B., and Nagpal, R. (2022). A hybrid pso algorithm for multi-robot target search and decision awareness. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11520–11527. 101, 110

[Elloumi et al., 2014] Elloumi, W., Treuillet, S., and Leconge, R. (2014). Real-time camera orientation estimation based on vanishing point tracking under manhattan world assumption. *Journal of Real-Time Image Processing*. 174

[Espressif, 2025] Espressif (2025). 'ESP-NOW Wireless Communication Protocol'. https://www.espressif.com/en/solutions/low-power-solutions/esp-now. 148, 154

[Eustice et al., 2005] Eustice, R., Singh, H., and Leonard, J. (2005). Exactly sparse delayed-state filters. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2417–2424. 62

[Furukawa et al., 2009] Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). Manhattan-world stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1422–1429. 174

[Ghanem et al., 2015a] Ghanem, B., Thabet, A., Carlos Niebles, J., and Caba Heilbron, F. (2015a). Robust manhattan frame estimation from a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3772–3780. 185, 186, 188

[Ghanem et al., 2015b] Ghanem, B., Thabet, A., Niebles, J. C., and Heilbron, F. C. (2015b). Robust manhattan frame estimation from a single rgb-d image. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3772–3780. 171, 172, 175

[Gielis et al., 2022] Gielis, J., Shankar, A., and Prorok, A. (2022). A critical review of communications in multi-robot systems. *Current Robotics Reports*, 3(4):213–225. 24

[globenewswire.com, 2024] globenewswire.com (2024). FireSwarm Solutions Inc. Secures $500,000 Investment from BC Centre for Innovation & Clean Energy to Advance Autonomous Drone Swarms for Wildfire Suppression. https://www.globenewswire.com/news-release/2024/12/11/2995519/0/en/FireSwarm-Solutions-Inc-Secures-500-000-Investment-from-BC-Centre-for-Innovation-Clean-Energy-to-Advance-Autonomous-Drone-Swarms-for-Wildfire-Suppression.html. 23

[Guo et al., 2019] Guo, R., Peng, K., Zhou, D., and Liu, Y. (2019). Robust visual compass using hybrid features for indoor environments. *Electronics*, 8(2):220. 184

[Halsted et al., 2021] Halsted, T., Shorinwa, O., Yu, J., and Schwager, M. (2021). A survey of distributed optimization methods for multi-robot systems. *CoRR*, abs/2103.12840. 24

[Handa et al., 2014] Handa, A., Whelan, T., McDonald, J., and Davison, A. (2014). A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China. 184

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107. 29

[Hartley and Kahl, 2007] Hartley, R. I. and Kahl, F. (2007). Global optimization through searching rotation space and optimal estimation of the essential matrix. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. 175

[Harvard University, 2014] Harvard University (2014). The Kilobot Project, Self-organizing Systems Research Group. https://web.archive.org/web/20141026212516/http://www.eecs.harvard.edu/ssr/projects/progSA/kilobot.html. 34

[He et al., 2020] He, M., Zhu, C., Huang, Q., Ren, B., and Liu, J. (2020). A review of monocular visual odometry. *The Visual Computer*, 36(5):1053–1065. 170

[Horn, 1984] Horn, B. (1984). Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686. 175

[Intel, 2024] Intel (2024). Intel/Realsense. https://www.intelrealsense.com/sdk-2/. 196

[Jamshidpey et al., 2025] Jamshidpey, A., Wahby, M., Allwright, M., Zhu, W., Dorigo, M., and Heinrich, M. K. (2025). Centralization vs. decentralization in multi-robot sweep coverage with ground robots and uavs. *Artificial Life and Robotics*. 24

[Jin et al., 2023] Jin, L., Zhang, J., Hold-Geoffroy, Y., Wang, O., Matzen, K., Sticha, M., and Fouhey, D. F. (2023). Perspective fields for single image camera calibration. *CVPR*. 195

[Jones and Hauert, 2024] Jones, S. and Hauert, S. (2024). Distributed spatial awareness for robot swarms. In *17th International Symposium on Distributed Autonomous Robotic Systems*. Springer, Cham. 35, 122

[Joo et al., 2019] Joo, K., Oh, T.-H., Kim, J., and Kweon, I. S. (2019). Robust and globally optimal manhattan frame estimation in near real time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(3):682–696. 175, 185, 189, 190

[Kantaros et al., 2021] Kantaros, Y., Schlotfeldt, B., Atanasov, N., and Pappas, G. (2021). Sampling-based planning for non-myopic multi-robot information gathering. *Autonomous Robots*, 45:1–18. 101

[Kapoutsis et al., 2021] Kapoutsis, A. C., Michailidis, I. T., Boutalis, Y., and Kosmatopoulos, E. B. (2021). Building synergetic consensus for dynamic gas-plume tracking applications using uav platforms. *Computers & Electrical Engineering*, 91. 25, 101

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Neural Information Processing Systems (NeurIPS)*. 172

[Kim et al., 2018] Kim, P., Coltin, B., and Kim, H. J. (2018). Indoor rgb-d compass from a single line and plane. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4673–4680. 184, 185, 189, 190

[Košecká and Zhang, 2002] Košecká, J. and Zhang, W. (2002). Video compass. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Computer Vision — ECCV 2002*, pages 476–490, Berlin, Heidelberg. Springer Berlin Heidelberg. 174

[Langer et al., 2022] Langer, F., Bae, G., Budvytis, I., and Cipolla, R. (2022). Sparc: Sparse render-and-compare for cad model alignment in a single rgb image. *arXiv preprint arXiv:2210.01044*. 172

[LaValle, 1998] LaValle, S. M. (1998). Rapidly-exploring random trees : a new tool for path planning. *The annual research report*. 29

[Le Ny and Pappas, 2009] Le Ny, J. and Pappas, G. J. (2009). On trajectory optimization for active sensing in gaussian process models. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 6286–6292. 101

[Lee et al., 2009] Lee, D. C., Hebert, M., and Kanade, T. (2009). Geometric reasoning for single image structure recovery. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2136–2143. 174

[Lee et al., 2021] Lee, J., Go, H., Lee, H., Cho, S., Sung, M., and Kim, J. (2021). CTRL-C: Camera calibration TRansformer with Line-Classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 195

[Lee and Yoon, 2015] Lee, J.-K. and Yoon, K.-J. (2015). Real-time joint estimation of camera orientation and vanishing points. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1866–1874. 171, 174

[Li et al., 2023] Li, H., Zhao, J., Bazin, J.-C., Kim, P., Joo, K., Zhao, Z., and Liu, Y.-H. (2023). Hong kong world: Leveraging structural regularity for line-based slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 176

[Li et al., 2025] Li, P., Liu, J., Wu, Y., and Zhou, L. (2025). Failure-aware multi-robot coordination for resilient and adaptive target tracking. 25

[Li et al., 2020a] Li, Q., Gama, F., Ribeiro, A., and Prorok, A. (2020a). Graph neural networks for decentralized multi-robot path planning. In *International Conference on Intelligent Robots and Systems (IROS)*. 78

[Li et al., 2020b] Li, Y., Brasch, N., Wang, Y., Navab, N., and Tombari, F. (2020b). Structure-slam: Low-drift monocular slam in indoor environments. *IEEE Robotics and Automation Letters*, 5(4):6583–6590. 172

[Li and Tombari, 2022] Li, Y. and Tombari, F. (2022). E-graph: Minimal solution for rigid rotation with extensibility graphs. In *European Conference on Computer Vision*, pages 306–322. Springer. 184, 185, 189, 190

[Lienert et al., 2019] Lienert, T., Stigle, L., and Fottner, J. (2019). Failure-handling strategies for mobile robots in automated warehouses. *33rd International ECMS Conference on Modelling and Simulation*. 23

[Liu et al., 2023] Liu, S., Fan, L., Johns, E., Yu, Z., Xiao, C., and Anandkumar, A. (2023). Prismer: A vision-language model with an ensemble of experts. *arXiv*. 172

[Liu and Lee, 2020] Liu, Y. and Lee, K. (2020). Probabilistic consensus decision making algorithm for artificial swarm of primitive robots. *SN Applied Sciences*, 2. 121, 139, 140, 143

[Luis et al., 2020] Luis, C., Vukosavljev, M., and Schoellig, A. (2020). Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, PP:1–1. 78, 79, 102

[Man et al., 2018] Man, Y., Weng, X., Li, X., and Kitani, K. (2018). Groundnet: Monocular ground plane normal estimation with geometric consistency. *Proceedings of the 27th ACM International Conference on Multimedia*. 196

[Matsuki et al., 2024] Matsuki, H., Murai, R., Kelly, P. H. J., and Davison, A. J. (2024). Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 37

[Morimoto and Chellappa, 1998] Morimoto, C. and Chellappa, R. (1998). Evaluation of image stabilization algorithms. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 5, pages 2789–2792. IEEE. 170

[Mukadam et al., 2018a] Mukadam, M., Dong, J., Dellaert, F., and Boots, B. (2018a). STEAP: simultaneous trajectory estimation and planning. *CoRR*, abs/1807.10425. 64

[Mukadam et al., 2018b] Mukadam, M., Dong, J., Yan, X., Dellaert, F., and Boots, B. (2018b). Continuous-time gaussian process motion planning via probabilistic inference. *International Journal of Robotics Research (IJRR)*, 37(11). 55, 78, 82, 83

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163. 171, 185, 189

[Murai et al., 2024a] Murai, R., Dexheimer, E., and Davison, A. J. (2024a). MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. *arXiv preprint*. 37

[Murai et al., 2024b] Murai, R., Ortiz, J., Saeedi, S., Kelly, P. H. J., and Davison, A. J. (2024b). A robot web for distributed many-device localization. *IEEE Transactions on Robotics*, 40:121–138. 28, 68, 79, 80, 95, 96, 102, 124, 146

[Murphy et al., 1999] Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the*

Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI'99, page 467–475, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 57

[Ortiz et al., 2021] Ortiz, J., Evans, T., and Davison, A. J. (2021). A visual introduction to Gaussian Belief Propagation. *arXiv preprint arXiv:2107.02308*. 72

[Parra Bustos et al., 2016] Parra Bustos, A., Chin, T.-J., Eriksson, A., Li, H., and Suter, D. (2016). Fast rotation search with stereographic projections for 3d registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2227–2240. 175

[Patwardhan, 2023] Patwardhan, A. (2023). 'GBP Planner Code'. https://aalpatya.github.io/gbpplanner. 149

[Patwardhan and Davison, 2024] Patwardhan, A. and Davison, A. J. (2024). A distributed multi-robot framework for exploration, information acquisition and consensus. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12062–12068. 40, 119, 122, 124

[Patwardhan and Davison, 2025] Patwardhan, A. and Davison, A. J. (2025). Dancers: A distributed algorithm for negotiating consensus in robot swarms with gaussian belief propagation. *arXiv preprint arXiv:2508.18153*. 40

[Patwardhan et al., 2023] Patwardhan, A., Murai, R., and Davison, A. J. (2023). Distributing collaborative multi-robot planning with gaussian belief propagation. *IEEE Robotics and Automation Letters*, 8(2):552–559. 39, 99, 102, 108, 122

[Patwardhan et al., 2025] Patwardhan, A., Rhodes, C., Bae, G., and Davison, A. (2025). U–ARE–ME: Uncertainty-aware rotation estimation in manhattan environments. In *International Conference on 3D Vision 2025*. 41

[Pautrat et al., 2023] Pautrat, R., Liu, S., Hruby, P., Pollefeys, M., and Barath, D. (2023). Vanishing point estimation in uncalibrated images with prior gravity direction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14118–14127. 171, 174, 185, 188

[Pearl, 1982] Pearl, J. (1982). Reverend bayes on inference engines: a distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, page 133–136. AAAI Press. 28, 56

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34. 34

[Rogers and Gross, 2013] Rogers, T. and Gross, T. (2013). Consensus time and conformity in the adaptive voter model. *Phys. Rev. E*, 88:030102. 24, 121

[Savkin et al., 2016] Savkin, A., Wang, C., Baranzadeh, A., Xi, Z., and Nguyen, H. (2016). Distributed formation building algorithms for groups of wheeled mobile robots. *Robotics and Autonomous Systems*, 75:463–474. 121

[Scheidler et al., 2016] Scheidler, A., Brutschy, A., Ferrante, E., and Dorigo, M. (2016). The k-unanimity rule for self-organized decision making in swarms of robots. *IEEE Transactions on Cybernetics*, 46:1175. 121, 139

[Schindler and Dellaert, 2004] Schindler, G. and Dellaert, F. (2004). Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 176

[Senbaslar et al., 2019] Senbaslar, B., Honig, W., and Ayanian, N. (2019). Robust trajectory execution for multi-robot teams using distributed real-time replanning. In *Proceedings of Distributed and Autonomous Robotics Systems*. 78

[Shan and Mostaghim, 2021] Shan, Q. and Mostaghim, S. (2021). Discrete collective estimation in swarm robotics with distributed bayesian belief sharing. *Swarm Intelligence*, 15(4):377–402. 121

[Shirsat et al., 2020] Shirsat, A., Elamvazhuthi, K., and Berman, S. (2020). Multi-robot target search using probabilistic consensus on discrete markov chains. In *2020 IEEE*

International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 108–115. 101

[Siemensma et al., 2024] Siemensma, T., Chiu, D., Ramshanker, S., Nagpal, R., and Haghighat, B. (2024). Collective bayesian decision-making in a swarm of miniaturized robots for surface inspection. In *Swarm Intelligence: 14th International Conference, ANTS 2024, Proceedings*, page 57–70. Springer-Verlag. 121

[Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*. 171, 172, 175, 185, 186, 188

[Solà, 2017] Solà, J. (2017). Quaternion kinematics for the error-state kalman filter. *CoRR*, abs/1711.02508. 64

[Sola et al., 2018] Sola, J., Deray, J., and Atchuthan, D. (2018). A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*. 183

[Soria et al., 2021] Soria, E., Schiano, F., and Floreano, D. (2021). Predictive control of aerial swarms in cluttered environments. *Nature Machine Intelligence*, 3:545–554. 77

[St-Onge et al., 2020] St-Onge, D., Varadharajan, V. S., Švogor, I., and Beltrame, G. (2020). From design to deployment: Decentralized coordination of heterogeneous robotic teams. *Frontiers in Robotics and AI*, 7. 25

[Straub et al., 2015] Straub, J., Bhandari, N., Leonard, J. J., and Fisher, J. W. (2015). Real-time manhattan world rotation estimation in 3d. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1913–1920. 171, 172, 185, 186, 188

[Straub et al., 2014] Straub, J., Rosman, G., Freifeld, O., Leonard, J. J., and Fisher, J. W. (2014). A mixture of manhattan frames: Beyond the manhattan world. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3770–3777. 175, 176

[Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. 184

[Sun et al., 2023] Sun, G., Zhou, R., Ma, Z., Li, Y., Groß, R., Chen, Z., and Zhao, S. (2023). Mean-shift exploration in shape assembly of robot swarms. *Nature Communications*, 14. 119, 121, 134, 135, 139

[Tatzgern et al., 2014] Tatzgern, M., Grasset, R., Kalkofen, D., and Schmalstieg, D. (2014). Transitional augmented reality navigation for live captured scenes. In *2014 IEEE Virtual Reality (VR)*, pages 21–26. 170, 196

[Taylor and Cowley, 2012] Taylor, C. and Cowley, A. (2012). Parsing indoor scenes using rgb-d imagery. In *Proceedings of Robotics: Science and Systems*. 175

[theengineer.co.uk, 2018] theengineer.co.uk (2018). Robot swarm runs hive of online grocery activity. https://www.theengineer.co.uk/content/in-depth/robot-swarm-runs-hive-of-online-grocery-activity. 23

[Valentini et al., 2017] Valentini, G., Ferrante, E., and Dorigo, M. (2017). The best-of-n problem in robot swarms: formalization, state of the art, and novel perspectives. *Frontiers in Robotics and Artificial Intelligence*, 4. 35, 119

[van den Berg et al., 2009] van den Berg, J., Guy, S., Lin, M., and Manocha, D. (2009). Reciprocal N-body collision avoidance. *Proceedings of the International Symposium on Robotics Research (ISRR)*. 24, 77, 87

[van den Berg and Overmars, 2005] van den Berg, J. and Overmars, M. (2005). Prioritized motion planning for multiple robots. In *International Conference on Intelligent Robots and Systems (IROS)*. 24, 77

[Van Parys and Pipeleers, 2016] Van Parys, R. and Pipeleers, G. (2016). Online distributed motion planning for multi-vehicle systems. In *Proceedings of the European Control Conference (ECC)*. 78

[Wang and Rubenstein, 2020] Wang, H. and Rubenstein, M. (2020). Shape formation in homogeneous swarms using local task swapping. *IEEE Transactions on Robotics*, 36(3):597–612. 122, 146

[Wessnitzer and Melhuish, 2003] Wessnitzer, J. and Melhuish, C. (2003). Collective decision-making and behaviour transitions in distributed ad hoc wireless networks of mobile robots: Target-hunting. In *Advances in Artificial Life*, pages 893–902. Springer Berlin Heidelberg. 121, 139

[Woosley et al., 2020] Woosley, B., Dasgupta, P., Rogers, J., and Twigg, J. (2020). Multi-robot information driven path planning under communication constraints. *Autonomous Robots*, 44(5):721–737. Publisher Copyright: © 2019, Springer Science+Business Media, LLC, part of Springer Nature. 100

[YouTube, 2018] YouTube (2018). 'Race The Tube - London Parkour POV'. https://www.youtube.com/watch?v=tXMPRK2LQAE. 200

[YouTube, 2019] YouTube (2019). 'Walking in the Rain Tokyo, Japan (Relaxing Binaural Thunderstorm Sounds for Sleep) 4k ASMR'. https://www.youtube.com/watch?v=Et7O5-CzJZg. 199

[YouTube, 2021] YouTube (2021). 'A Walk Around Orison City - Star Citizen Alpha 3.14 gameplay (no commentary)'. https://www.youtube.com/watch?v=G3gqBaqDSaE. 199

[Zhai et al., 2023] Zhai, G., Huang, D., Wu, S.-C., Jung, H., Di, Y., Manhardt, F., Tombari, F., Navab, N., and Busam, B. (2023). Monograspnet: 6-dof grasping with a single rgb image. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1708–1714. IEEE. 172

[Zhang et al., 2025] Zhang, S., Li, Z., Yin, Y., Xu, S., Chaudhary, V., and Xu, H. (2025). A survey on multi-robot collaboration systems: Architectures, performances, and applications. 24

[Zhang et al., 2022] Zhang, Y., Ding, Y., Song, J., Li, J., and Wei, H.-L. (2022). A fast manhattan frame estimation method based on normal vectors. *Journal of Field Robotics*, 39(5):557–579. 175, 177, 179

[Zhang et al., 2024] Zhang, Y., Zhou, R., Li, X., and Sun, G. (2024). Mean-shift shape formation of multi-robot systems without target assignment. *IEEE Robotics and Automation Letters*, 9(2):1772–1779. 122

[Zheng and Lee, 2023] Zheng, C. and Lee, K. (2023). Consensus decision-making in artificial swarms via entropy-based local negotiation and preference updating. *Swarm Intelligence*, 17:1–21. 119, 121, 139, 143

[Zhou et al., 2021] Zhou, X., Zhu, J., Zhou, H., Xu, C., and Gao, F. (2021). Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4101–4107. 102