Imperial College London
Department of Computing

Distributed Spatial AI

Riku Murai

25th February 2025

Supervised by Professor Paul H.J. Kelly, 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.

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 *Distributed Spatial AI* – how devices can coordinate to perform inference in a decentralised manner to enhance their Spatial AI abilities beyond each device's perceptual capabilities while achieving low-power and scalability.

First, we investigate factorised computation, which aims to minimise the cost of data transfer by collocating the processing near the sensor where the data is captured. We develop a Visual Odometry (VO) system using a focal-plane sensor-processor, which enables computations such as feature extraction to occur directly on the camera's focal-plane. This enables our VO pipeline to operate at 300 FPS, making it robust against violent motions while also being low-power. We then explore the scene representation for visual Simultaneous Localisation and Mapping (SLAM). Representing the scene with many 3D Gaussian blobs, we achieve near-photorealistic fidelity 3D reconstruction online using a single moving camera. All the images captured by the camera are compressed into a single 3D representation from which we can re-render the images at near original quality and perform novel-view synthesis between views.

The core of this thesis is the investigation of how scalable, accurate, and robust many-device localisation can be achieved. We argue that Gaussian Belief Propagation (GBP) is a promising algorithmic candidate for Distributed Spatial AI, and using GBP, we develop *Robot Web*, a framework for decentralised localisation. Extending the formulation of GBP to support Lie groups, we demonstrate GBPs ability to localise 1000s of devices even under challenging situations, such as communication failures and large amounts of outlying measurements, using only ad-hoc peer-to-peer communication. The asynchronous property of GBP enables the definition of a simple communication protocol, which individual devices can implement to participate in co-localisation. Finally, we enhance the Robot Web framework to enable autocalibration of the sensors' and markers' extrinsic while simultaneously performing localisation, further improving the accuracy of localisation.

Acknowledgements

My sincere thanks go to both Professor Andrew J. Davison and Professor Paul H.J. Kelly for their support throughout my Ph.D. These four years have been invaluable in helping me grow, both academically and personally.

Andy has shown me the importance of dedicating time to solving interesting and meaningful problems. Even when our research did not progress as smoothly as we had hoped, his trust in our work encouraged us to persist. His willingness to learn and explore diverse areas of research is something I hope to emulate in my own career. I have learnt a lot from the numerous conversations I had with Paul. His knowledge always seemed limitless, and from him, I've learnt the value of exploring topics from a wide range of fields and maintaining a positive attitude towards research. I am also deeply grateful to both my Ph.D. examiners Professor Luca Carlone and Professor Christos Bouganis for all the discussions which helped me improve my thesis.

This journey would not have been possible without the support of my colleagues and friends. I'd like to thank Iosifina Pournara for ensuring everything ran smoothly throughout. I have learned so much by collaborating with everyone in the lab, and none of this work would have been achievable without their help. I am thankful to Professor Sajad Saeedi, Professor Stefan Leutenegger, Joe Ortiz, Raluca Scona, Edgar Sucar, Tristan Laidlow, and Kentaro Wada for their support during the early stages of my Ph.D. and especially through the COVID period. I am particularly thankful to Hide Matsuki, Eric Dexheimer, Ignacio Alzugaray-Lopez, Aalok Patwardhan, and Edward Stow for all the fun research discussions and for pulling late-nighters to meet our paper deadlines. Finally, I'd like to thank Kirill Mazur, Marwan Taher, Xin Kong, Ivan Kapelyukhl, Seth Nabarro, Shikun Liu, Anagh Malik, Dorian Hennings, Gwangbin Bae, Callum Rhodes, George Bisbas, Luke Panayi. I feel fortunate that everyone in the lab feels more like a friend than just a colleague and I truly enjoyed coming to the lab every day and having conversations over coffee or a pint. The memories created during some of our trips will stay with me for the rest of my life.

I'm fortunate to have interned with Meta during my Ph.D. I'd like to thank all my colleagues, especially Luis Pineda, Mustafa Mukadam, Chris Paxton, and Robert Gieselmann, for making my time there fun and interesting.

Finally, I would like to thank my family for their support, especially during the challenges of COVID. Without their understanding and encouragement, completing my Ph.D. would not have been possible.

Contents

CHAPTER 1

Introduction

Contents

Not only is modern computing technology faster, more affordable, and more compact compared to the ENIAC, the first general-purpose computer built in 1945 – which weighed over 27 tonnes and consumed 150 kW for just 500 FLOPS – but in less than a century, computing devices have woven themselves into the fabric of our everyday lives. Looking forward, we envision a near future where devices with computational and sensing capabilities are so ubiquitous that they seamlessly integrate into the background [Weiser, 1991]. To achieve such a vision, the sophistication of these devices must go beyond mere computational power; they must understand and interact with the physical world around them.

1.1 Spatial AI

Spatial AI is an online problem where incoming data is processed in real-time to enable devices to usefully and meaningfully interact with their surrounding environments [Davison, 2018].

At the core of the Spatial AI system is perception. Devices must in real-time perceive and update the understanding of the surrounding world. Defining the exact meaning of understanding is challenging, and here, we define such ambiguous capability to be the act of building an internal representation necessary to usefully interact with the 3D environment. A robot might pick up an empty coke can to trash, fold up laundry, or vacuum the room. Whatever the task is, it must explicitly or implicitly build a model which informs the robot whether the next sequence of planned action is meaningful. Such a predictive capability, or a world model – estimating how the state of the world would change given the current state and the action – is necessary for Spatial AI to be truly useful. Ideally, such internal representation should be grounded in language, which opens up a natural interface between us and robots. This motivates the need to embed semantic or language features into the 3D map reconstructed by a SLAM system. Building them under both real-time and power constraint is non-trivial and potentially require innovation and co-optimisation of a full stack, from physical embodiment, software, and processor/sensor design. In this thesis, we primarily focus on *localisation*, the most fundamental, and starting point of any Spatial AI problem.

1.1.1 Low Power and Low Latency Sensing

Despite significant advances in Spatial AI, with Simultaneous Localisation and Mapping (SLAM) and Visual-Inertial Odometry (VIO) now providing localisation to centimetre accuracy, a considerable gap exists between real-world systems' needs and what the current state-of-the-art Spatial AI systems can offer. For instance, Augmented Reality (AR) glasses should continuously log all events experienced by the wearer throughout the day, allowing for later retrieval of any specific entries. Such capability requires the glasses to perform *always-on-sensing*, consistently monitoring the surrounding environment and tracking the position of the glasses to detect and record events autonomously. All these functionalities must be achieved while keeping the total weight of the glasses below 40g to ensure wearer comfort [Kim et al., 2021].

In terms of energy consumption, current Spatial AI systems are far from meeting such demands. For example, the capacity of a modern phone's battery (*e.g.* iPhone 15) is around 13Wh, and a modern GPU often used for recent SLAM systems (*e.g.* Nvidia RTX 4090) draws up to around 450W. Even if we ignore all the other components, such as the CPU, fans, and *etc.*,it can only operate for approximately 100 seconds! Assuming we are awake for about 15 hours a day on a phone's battery, to achieve 15 hours of active use, the system can only draw 0.87W in total, which is $1/520$ of an RTX 4090's power draw. Moreover, the battery capacity is even more limited for wearable glasses such as Project Aria [Somasundaram et al., 2023]. With only 2.5Wh for 15 hours of operation, the system can only draw 0.17W, $1/2700$ of RTX 4090s' consumption. We are limited to this budget unless we expect a significant leap in the battery technology.

Minimising the Spatial AI system's end-to-end latency is also essential. When the surrounding environment changes, robots must quickly adapt their planned actions to avoid accidents, and any reduction in system latency provides a wider window for these adjustments. Additionally, low

latency benefits agile robotics; for example, visual processing required to catch a ball becomes trivial if a system can provide high-speed visual feedback [Murakami et al., 2015]. Moreover, for AR / Virtual Reality (VR) applications to feel seamless and natural, it is recommended that the end-to-end latency is kept below 20ms [Stauffert et al., 2020]. Reducing the latency and operating at high speed is often beneficial (*e.g.* for camera pose optimisation) [Handa et al., 2012]; however, the higher frame rate increases the volume of data that must be transferred and processed, and thus consumes more energy.

The camera used for SLAM alone consumes 0.5-2W, already exceeding the power budget that an always-on wearable device can afford. These cameras also produce a large amount of data; for instance, recording full-HD frames (1920×1080) at 100 FPS generates around 0.2GB/sec, and eventually, processing and handling such a large amount of information becomes an issue [Martel, 2019]. This clearly illustrates a situation where the conventional approaches fall short, motivating us to investigate *unconventional* approaches for processing visual information, such as performing pixel-parallel processing on the camera's focal-plane.

1.1.2 Collaborative Sensing

As our environment becomes populated with intelligent devices equipped with computing capabilities, these devices will operate – and possibly collaborate – within the same shared space.

Currently, Spatial AI systems assume that individual devices operate independently, which inherently limits their spatial understanding to their own perceptual capabilities. The absence of knowledge about other devices' observations and intentions leads to suboptimal spatial understanding, so as the number of devices operating increases, even a simple task, such as robots navigating around each other, becomes a challenge.

Whatever we cannot observe, we must infer. To compensate for the absence of observations, we often rely on a data-driven prior, such as Deep Learning [Goodfellow et al., 2016], which has seen widespread success in various fields over recent years. However, the effectiveness of these approaches, particularly in 3D vision and robotics (and even more so in multi-robot systems), is primarily limited by the lack of availability of high-quality data. Thus, as of now, the prospect of developing a general-purpose *Spatial AI prior* capable of compensating for any missing spatial information remains impractical. Nevertheless, relying solely on data from a single device without any data-driven prior is severely limiting due to the incompleteness of the observations, and the limited number of observations makes the system less robust and prone to failures.

Fortunately, we are not constrained to an environment with a single device in most realistic scenarios. Instead, we have abundant sensing and computing capabilities scattered throughout our environment. These devices are equipped with various sensing capabilities, potentially reinforcing, rejecting outliers, or revealing new information when the observations are jointly considered. Hence, the question we ask is: *How can we effectively cooperate and coordinate multiple devices to create an accurate and robust, yet scalable Spatial AI system?*

1.2 Distributed Spatial AI

One clear possibility of performing inference jointly using all the available data is to upload the captured information to a cloud. While this centralised approach is the simplest and possibly the most accurate, it has many drawbacks, including the need for centralised authority and the requirement to share potentially sensitive information, making such an approach unattractive for widespread adoption. Instead, we explore an alternative direction where our system is decentralised and operates only by peer-to-peer communication.

In a Distributed Spatial AI system, computation must be *asynchronous*. This allows for simultaneous operations across the network of devices without the need for synchronisation and ensures that the network's performance is not bottlenecked by less capable devices, thereby enhancing overall efficiency and responsiveness. Moreover, asynchronicity enables flexible communication methods, such as multi-hop, which allows for a much more efficient network/architecture design (*e.g.* on a connected graph, messages can be directed to and from any nodes without altering the connectivity).

The system must support *heterogeneous* devices. As the system scales, different devices will have distinct processing and sensing capabilities. Heterogeneity requires the system to fuse multiple observations with various modalities and precisions; this motivates a *probabilistic* approach, for example, Bayesian inference.

In a distributed system, failures are unavoidable, and we require the algorithm to be *robust*. For example, when a device moves, it connects and disconnects from the other devices, changing the network's topology. Furthermore, communication is not always reliable and can fail or be delayed, and just like any other system, it must be robust against outlying measurements.

In a Distributed Spatial AI system, we require asynchronicity, heterogeneity, and robustness. To achieve a scalable system, a probabilistic formulation, especially probabilistic graphical models, is useful and provides the right abstraction. A global problem can be represented as a graph, and a Distributed Spatial AI system can split the global graph into fragments and let each device locally own its subgraph. By exchanging messages between devices using a unified probabilistic protocol, the problem can be solved in a decentralised manner. Using a message-passing algorithm with a straightforward inter-device interface simplifies the overall system, which is essential for scalability.

1.3 Probabilistic Inference

In this section, we discuss some probabilistic methods commonly used to solve Spatial AI problems. The Bayesian principle offers a systematic method for fusing multiple uncertain informa-

tion. In Bayesian inference, we infer the posterior distribution using Bayes' Theorem:

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} , \tag{1.1}$$

where the posterior is $p(\mathbf{x}|\mathbf{z})$, the conditional distribution over the state $\mathbf{x}$ given some observations $\mathbf{z}$. This posterior can be perceived as our updated belief about the world's state [Murphy, 2012, Chapter 3], and our belief is iteratively refined and updated by incorporating new evidence $\mathbf{z}$. Thus, using Bayes' Theorem enables a progressively more accurate understanding of the system's state from uncertain data.

The posterior summarises everything which can be inferred from $\mathbf{z}$ about the unknown state $\mathbf{x}$ [Murphy, 2012, Chapter 5]. Here, we give two summary statistics which are often used in Spatial AI:

Maximum a Posteriori (MAP) Inference is a point-estimate of the most likely configuration of the states given the observations. MAP is found by finding a configuration which maximises the posterior distribution:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) \tag{1.2}$$

$$= \arg\max_{\mathbf{x}} p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) \tag{1.3}$$

$$= \arg\max_{\mathbf{x}} l(\mathbf{x};\mathbf{z})p(\mathbf{x}) . \tag{1.4}$$

Here, $l(\mathbf{x};\mathbf{z})$ is the likelihood of the state $\mathbf{x}$ given the observations $\mathbf{z}$, and is defined as a function proportional to $p(\mathbf{z}|\mathbf{x})$:

$$l(\mathbf{x};\mathbf{z}) \propto p(\mathbf{z}|\mathbf{x}) . \tag{1.5}$$

This notation clarifies that the likelihood is a function of $\mathbf{x}$, and $\mathbf{z}$ is the function parameter [Dellaert and Kaess, 2017].

Notice that in Equation 1.4 instead of maximising, we can equivalently minimise the negative log-probability:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} -\log p(\mathbf{x}|\mathbf{z}) , \tag{1.6}$$

$$= \arg\min_{\mathbf{x}} -\log l(\mathbf{x};\mathbf{z}) - \log p(\mathbf{x}) . \tag{1.7}$$

The first term is the negative log-likelihood, and the second term is the negative log-prior. If the underlying distribution of the likelihood and the prior is Gaussian, then the negative log-likelihood is equivalent to the least-squares objective function, and the negative log-prior acts as a regularisation term [Deisenroth et al., 2020, Chapter 9]. In many Spatial AI problems such as pose-graph optimisation [Kümmerle et al., 2011], bundle adjustment [Triggs et al., 1999], and visual SLAM [Strasdat et al., 2012], the observations and priors are assumed to follow a Gaussian noise and the problem is solved using efficient non-linear least squares solvers.

Marginal Inference not only estimates the most likely configuration of the state but also provides the uncertainty for each of the estimated states. The joint posterior distribution is marginalised

over the variables of interest, which is defined as (assuming discrete variables):

$$p(x_i) = \sum_{\mathbf{x}/x_i} p(\mathbf{x}|\mathbf{z}) \,, \tag{1.8}$$

summing over all the variables $\mathbf{x}$ except x_i. Compared to MAP inference which *loses* information, marginal inference retains the uncertainty, which can be useful for decision-making tasks [Deisenroth et al., 2020, Chapter 8].

A Factor Graph is a graphical model representing the probabilistic problem's structure and factorisation. The factorised representation reveals the locality of the problem, which the inference algorithms can exploit. A factor graph is a bipartite graph $G = (X, F, E)$ consisting of variable nodes $X = \{x_i\}_{i=1:N_v}$ connected by edges E to factor nodes $F = \{f_s\}_{s=1:N_f}$. A factor graph represents the factorisation of the joint distribution:

$$p(\mathbf{x}) \propto \prod_{s=1}^{N_f} f_s(\mathbf{x}_s) \,, \tag{1.9}$$

where $\mathbf{x}_s = n(f_s)$. Here, $n(x)$ is the set of neighbouring nodes connected via edge to the node x.

Typically for our application, we use a factor graph to factorise the joint posterior $p(\mathbf{x}|\mathbf{z})$ distribution, and for the factors, we use likelihood functions $f(\mathbf{x}_s) = l(\mathbf{x}_s; \mathbf{z}_s)$. Not only do factor graphs reveal exploitable locality, but they also provide a clear and intuitive approach to map out the problem in an interpretable manner, which, for example, is useful for communicating one's idea [Dellaert and Kaess, 2017].

1.4 Gaussian Belief Propagation

Belief Propagation (BP), as initially introduced by Judea Pearl in the early 1980s, is a seminal message-passing algorithm to efficiently perform marginal inference on tree-structured graphical models [Pearl, 1982]. BP computes the exact marginals on a tree. However, the algorithm's applicability to a more general graph (which includes cycles) was unclear. Empirically, [Murphy et al., 1999] demonstrated that Loopy Belief Propagation (LBP) – applying BP iteratively on a general graph – is a good approximation to the true marginal posterior when the algorithm converges. With further theoretical analysis of LBP, it was shown that the mean estimate of LBP at convergence is equal to the stationary point of a Bethe approximation of the free energy [Yedidia et al., 2001]. This insight provided a theoretical underpinning for the empirical observations and significantly broadened the understanding of LBP's applicability and the conditions under which it offers accurate approximations.

LBP is a local algorithm that operates by message-passing between the variable and factor nodes. One of the remarkable characteristics of LBP is that it can converge without the need for any *global information*, meaning that nodes are unaware of the broader problem structure beyond its adjacent nodes. This local operation paradigm is particularly advantageous for distributed computing. It allows LBP to scale efficiently across many devices as it eliminates the need for

any global synchronisation, which is slow and possibly impossible in many large-scale real-world applications.

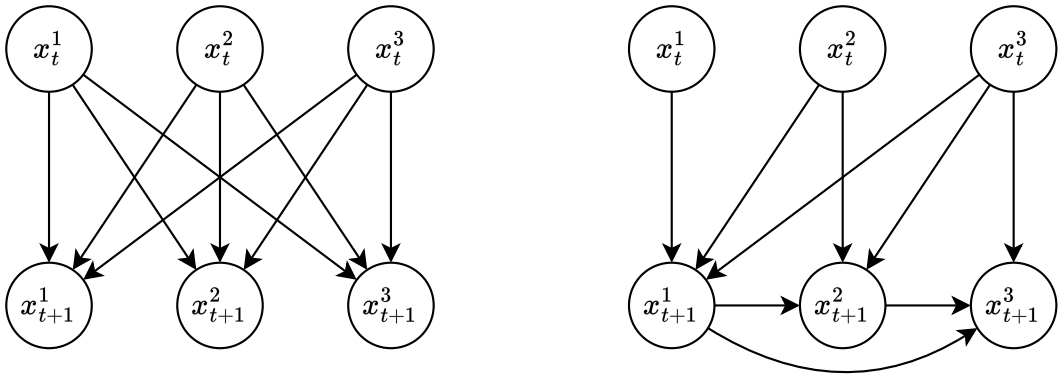
LBP's flexible message-passing schedules are favourable from a distributed systems perspective. It can converge under various message-passing schedules, where nodes exchange messages in no specific order, and the absence of a requirement for global coordination among nodes makes the algorithm robust, especially against delayed or failed communications.

Gaussian Belief Propagation (GBP) is a variant of BP that operates on Gaussian factor graphs, where the factors and the variables are Gaussians. The Gaussianity assumption allows all operations in BP to be efficient with a closed-form solution and also enjoys stronger correctness guarantees compared to LBP, where the inferred point-estimate is *exact* upon convergence [Weiss and Freeman, 1999]. GBP inherits all the aforementioned properties of BP, maintaining its scalability and robustness, which are essential for applications across a broad spectrum of domains. Despite the complexity of its theoretical proofs regarding convergence, GBP remains remarkably straightforward from an operational standpoint. BP has been applied widely across many applications and fields, with its most notable success in error-correcting codes. These codes are designed to transmit data through noisy channels by incorporating redundancy, allowing the receiver to correct any errors encountered. To ensure bandwidth efficiency, the redundancy in error-correcting code must be kept to a minimum. Turbo codes represent the first practical implementation of codes nearly achieving the Shannon Limit, or maximum channel capacity [Berrou et al., 1993]. Initially, the connection between Turbo codes and LBP was not clear, and it was only later discovered that the decoding algorithm used in turbo codes is fundamentally equivalent to LBP [McEliece et al., 1998].

More related to Spatial AI, BP has been applied to many vision problems such as optical flow and stereo matching [Sun et al., 2003, Felzenszwalb and Huttenlocher, 2006]. Expanding into the domain of SLAM, LoopySAM [Ranganathan et al., 2007] leverages GBP for efficient and incremental SLAM, using a wild-wire algorithm, where only the nodes with large updates are computed.

More recently, GBP has been applied to challenging 3D vision problems such as Bundle Adjustment where 3D map points and camera poses are jointly optimised [Ortiz et al., 2020], incremental planar abstraction of the 3D map points [Ortiz et al., 2022], predicting 6D scene flow [Scona et al., 2022], and efficiently mapping gas concentration in an indoor environment [Rhodes et al., 2022].

Beyond single-device applications, BP has also been applied to multi-device optimisation. For example, non-parametric BP [Schiff et al., 2009] or a hybrid combination of parametric and non-parametric BP [Wan et al., 2017] is used for localisation, leveraging the distributed nature of BP to optimise across devices. GBP has also been applied for applications such as multi-robot motion planning [Patwardhan et al., 2023] and exploration [Patwardhan and Davison, 2023]. Additionally, BP is a local algorithm and can be used for near-sensor processing. This opens up interesting opportunities such as low-power, event-based computation [Nagata and Sekikawa, 2023].



(a) Jacobi (b) Gauss-Seidel

Figure 1.1: Dependency graph for one iteration of Jacobi and Gauss-Seidel iterations.

1.5 Related Algorithms for Distributed Inference

This section discusses some of the algorithms related to GBP. Similar to GBP, these approaches perform distributed inference in parallel across many processors or devices.

Distributed Gradient Descent Performing gradient descent on a factor graph can be viewed as message passing, where every factor node locally computes a gradient for each adjacent variable using the variable's state, and these gradients are then averaged at every variable node. These operations can be fully parallelised and are suitable for distributed optimisation. They work even if the gradients are *stochastic*, making them asynchronous and robust against communication failure. Since gradient descent is a first-order method, their convergence rate is slow; however, by applying acceleration techniques such as preconditioning [Olson et al., 2006, Grisetti et al., 2007], it can be used to solve a large Pose Graph Optimisation (PGO) problem.

For distributed optimisation, [Knuth and Barooah, 2013] used distributed gradient descent on the Riemannian manifold for collaborative multi-robot localisation. In [Calafiore et al., 2010, Calafiore et al., 2012], gradient descent is used to localise the agents using range measurements in a decentralised manner. In [Tian et al., 2021, Tian et al., 2020], a block coordinate-descent on a Riemannian manifold is used to solve distributed PGO problem with *certifiable correctness* and robustness to asynchronous communication. Similarly, the Majorisation Minimisation (MM) method is used to solve distributed PGO [Fan and Murphey, 2020] and distributed Bundle Adjustment (BA) [Fan et al., 2023]. To improve convergence rate, [Tian et al., 2021, Tian et al., 2020, Fan and Murphey, 2020, Fan et al., 2023] uses Nesterov's Accelerated Gradient [Nesterov, 1983].

Compared to gradient descent, the convergence rate of GBP is typically faster since messages are a Gaussian distribution, which is more informative than the steepest direction of descent.

Distributed Jacobi and Gauss-Seidel Jacobi and Gauss-Seidel are parallel iterative methods used for solving systems of linear equations. In a distributed setting, we have many nodes, each computing a component of $\mathbf{x}$, a vector we are solving for. The main difference between the methods is that the components are computed in parallel in the Jacobi method, and $\mathbf{x}$ is updated simultaneously. However, in the Gauss-Seidel method, components of $\mathbf{x}$ are computed in a sweep, updated one at a time, and the most recently computed value is used to compute the other components [Bertsekas and Tsitsiklis, 2015], as depicted in Figure 1.1. Gauss-Seidel converges significantly faster than the Jacobi method; however, parallelising is less trivial. Empirically, it has been demonstrated that GBP converges faster than Jacobi, Gauss-Seidel, and their relaxations [Bickson, 2008].

In the context of distributed Spatial AI, the Jacobi method is used by [Barooah and Hespanha, 2005] and [Aragues et al., 2011] for multi-device localisation. Both Gauss-Seidel and Jacobi methods are used in [Choudhary et al., 2017], where distributed PGO is solved using chordal initialisation [Martinec and Pajdla, 2007, Carlone et al., 2015b], and this approach is used as a backend for multi-robot SLAM methods [Lajoie et al., 2020, Cieslewski et al., 2018]

Alternating Direction Method of Multipliers The Alternating Direction Method of Multipliers (ADMM) algorithm solves optimisation problems by partitioning the problem into small pieces, allowing them to be solved in a distributed manner. The ADMM belongs to a class of algorithms called the method of multipliers. The method of multipliers solves a constrained optimisation problem by converting it into an augmented Lagrangian function – a Lagrangian function with a quadratic penalty term. The constrained optimisation problem is solved using dual ascent, which alternates between minimising the primal problem and performing gradient ascent on the dual problem. Here, the primal problem minimises the objective function while considering the current penalty of violating the optimisation constraints, and the dual problem solved via the gradient ascent step further enforces the penalties on the currently violated constraints. In ADMM, the primal problem is partitioned, such that they can be solved in a distributed fashion, in Gauss-Seidel-type iteration where we solve for each subproblem in a sweep, passing on the most up-to-date solutions [Boyd et al., 2011].

Typically, ADMM requires a centralised node to perform the gradient ascent on the dual variable. However, methods such as Consensus ADMM (C-ADMM) [Mateos et al., 2010] allow the dual variables to be updated locally. ADMM is used widely for distributed bundle adjustments [Eriksson et al., 2016, Zhang et al., 2017, Bänninger et al., 2023] as well as distributed PGO [Choudhary et al., 2015, McGann et al., 2023]. ADMM requires careful tuning of the penalty term, and also, the ascent step introduces an additional computation, which often must occur on a centralised node synchronously. Furthermore, as the constraints are gradually enforced, they can be slow to converge.

Gaussian Elimination Gaussian Elimination is used in frameworks such as GTSAM [Dellaert, 2012] and can be used for distributed inference, where the devices exchange Gaussian marginals.

In DDF-SAM [Cunningham et al., 2010] and DDF-SAM2 [Cunningham et al., 2013], Gaussian marginals about the co-observed variables are shared. While Gaussian Elimination can operate on any factor graph, as the cost of sharing the Gaussian marginal is quadratic in the number of variables, a sparsification method such as [Lazaro et al., 2013] is required to minimise communication. Furthermore, as Gaussian Elimination occurs on a linearised problem, linearisation points across devices must be consistent, which requires complex bookkeeping.

Federated Learning Federated learning is a machine learning technique designed to train models without the need to centrally collect raw user data, thereby preserving privacy [McMahan et al., 2017]. This restriction motivates the use of distributed algorithms for model training. Each device computes an update locally using its on-device user data, and only these updates are transmitted to a central server for global aggregation. Compared to standard machine learning settings, federated learning algorithms aim to address challenges such as minimising the number of communication rounds required and handling unbalanced data distribution across users.

1.6 Data Locality and Near-Sensor Processing

With the end of Moore's Law, historically predicting the doubling of a processor's transistor count approximately every two years, relying solely on improvements in processing capacity is no longer viable. Instead, attention has shifted first towards parallel [Sutter, 2005], but now towards heterogeneous computing, a concept explored in "Welcome to the Jungle" [Sutter, 2011]. Although the paper primarily concentrates on general computation, its principles apply equally to Spatial AI. As we push Spatial AI systems to operate at much lower power, the bottleneck increasingly becomes the energy consumption of the sensors and the data transfer from them. Hence, to minimise the data transfer, the computation must move closer to the sensor to perform data compression on the sensor itself.

Before we discuss the near-sensor processing, we first discuss the relationship between SLAM and the availability of new hardware.

1.6.1 Software-Hardware Co-design in SLAM

SLAM is a key component of Spatial AI, which aims to estimate the location and the map of the surrounding environment simultaneously in real time. It demands efficiency as real-time constraints impose strict limits on computational resources. Many breakthroughs are made by exploring and adopting new computing paradigms that can offer more efficient processing capabilities.

MonoSLAM [Davison et al., 2007] was the pioneering monocular SLAM system. It performs filtering of both camera poses and landmark positions and maps a small set of carefully chosen landmarks. Following this, PTAM [Klein and Murray, 2007] enhanced accuracy and reliability by tracking a larger amount of landmarks and implementing keyframe-based bundle adjustments.

PTAM performs real-time tracking by separating tracking and mapping into different threads, efficiently exploiting the capabilities of multi-core CPUs.

SLAM then gradually evolved from sparse points to dense maps, and such development was enabled by two key hardware: GPUs and depth cameras. Unlike the sparse set of points, the dense map captured the surface of the objects in the scene, enabling the map to be *interactive*, for example, by running a physics simulation or overlaying a AR object with sharp occlusion boundaries. In a dense visual SLAM system, photometric errors of every pixel are minimised for tracking and mapping. This is a significant amount of computation – sparse feature-based SLAM tracks against a couple thousand of 3D points [Klein and Murray, 2007], whereas every image contains over 30 thousand pixels even at VGA resolution. To process such a large amount of data, GPU's massively parallel processing capability was exploited for SLAM, enabling a live dense reconstruction using a monocular camera [Newcombe et al., 2011b]. Another advancement in dense SLAM can be attributed to the commodification of depth cameras. At 30 FPS, Microsoft's Kinect sensor captured both RGB and depth, and such measurements made dense SLAM not only simpler but much more robust. KinectFusion [Newcombe et al., 2011a] was the first SLAM system to utilise the Kinect sensor and depth camera since then is used for most dense 3D SLAM systems [Salas-Moreno et al., 2013, Whelan et al., 2015b, Dai et al., 2017].

Deep learning has also contributed significantly to SLAM. Priors such as depth [Tateno et al., 2017] and semantics [McCormac et al., 2017] were added to enhance robustness and versatility. In terms of tracking accuracy, DROID-SLAM [Teed and Deng, 2021] achieves state-of-the-art performance across many SLAM benchmarks by training a learnable SLAM pipeline with a differentiable Gauss-Newton optimiser. To further enhance the applicability of SLAM to more domains, for example, planning, a semantic segmentation network and a VIO system capable of recovering a dense mesh [Rosinol et al., 2020] was combined to produce a SLAM system, Hydra [Hughes et al., 2022], which is capable of creating a 3D scene graph – a hierarchy of representations such as places, objects, and the actual mesh – on the fly. All the mentioned SLAM systems utilise a GPU for fast inference and operate in real-time.

All the methods use images captured by the camera to perform SLAM. However, images are designed for human to look back at and capturing images at such high quality may be redundant for machine vision. This led to investigation of event cameras, which are bioinspired sensors that output an asynchronous stream of intensity changes. They offer attractive properties such as high temporal resolution, high dynamic range, and low energy consumption [Gallego et al., 2020]. Unlike conventional cameras, event cameras only output events asynchronously; hence, new algorithmic designs were required to process such data for Visual Odometry (VO)/SLAM. Each event does not contain sufficient information to fully constrain a camera's 6 Degrees of Freedom (DoF) motion. Thus many works use probabilistic filters which get updated with the asynchronous events [Kim et al., 2014, Censi and Scaramuzza, 2014, Kim et al., 2016, Gallego et al., 2017]. Semi-dense VO method, EVO [Rebecq et al., 2016], performs parallel tracking and mapping of the edges detected by the event camera and tracks robustly even under aggressive motion – benefiting from the advantageous properties of the event camera.

As *new hardware* are made available to the SLAM community, we've repeatedly observed that new capabilities are unlocked as such hardware gets integrated into a SLAM pipeline. However, too often, it is the hardware evolution which dictates the improvements of SLAM system – similar to the "hardware lottery" [Hooker, 2020] in Deep Learning. Projects such as Navion [Suleiman et al., 2019] instead explore the potential of *co-designing* the hardware and the algorithm together. They design and fabricate a stereo VIO Application-Specific Integrated Circuit (ASIC), which only consumes 2mW, a magnitude lower than the power dissipation of embedded CPUs.

Another interesting example is GraphCore's Intelligence Processing Unit (IPU) – a graph processor with Single Instruction Multiple Data (MIMD) architecture – which was successfully utilised for bundle adjustment [Ortiz et al., 2020], performing 24x faster than the state-of-the-art CPU implementation. The algorithm used, Gaussian Belief Propagation, is an example of an algorithm which *lost* in the hardware lottery and is typically magnitudes slower than the state-of-the-art bundle adjustment solvers on CPU or GPU. However, the IPUs have fast inter-tile (inter-core) communication, and the fast on-tile memory fits the computational pattern of GBP, demonstrating that when an algorithm and hardware match, there are significant performance gains to be expected.

Mobile devices require efficient, always-on Spatial AI systems as discussed in Section 1.1.1, and clearly, many technical innovations are needed to bridge the gap between the current state-of-the-art and the actual demands. Co-designing hardware and software, whilst challenging, has demonstrated magnitudes of potential improvements over the current systems. This motivates us to explore new unconventional hardware and design algorithms suitable for future chips and hardware.

1.6.2 Data Movement Bottlenecks

Data movement consumes non-trivial energy, and many works aim to minimise the data transfer distance by moving the computation closer to where the data is captured [Sze et al., 2017]. When a camera observes the surrounding world, a continuous visual data stream is transferred from a sensor to the processor. Moving data is not without cost; even converting an analog signal from the camera's photodiodes into a digital format consumes significant energy, and additionally, the energy consumption is proportional to the volume of data and the distance it travels. Hence, we want to minimise the data we digitise and then transfer.

To find an algorithm/hardware suitable for low-power/high-speed operation, we focus on *data locality*. In the next section, we discuss how we minimise the data transfer and thus the energy consumption by collocating a photodiode and a processor on the same pixel.

Figure 1.2: Example applications of SCAMP-5 FPSP. **Top Row:** Depth from focus [Martel et al., 2017]. Using a focus-tunable lens, images at different focuses are captured (left), and the sparse depth points (middle-right) are recovered by analysing the sharpness of each image using SCAMP-5. The right-most image is a post-processed, densified depth image. **Bottom Row:** HDR tone mapping [Martel et al., 2016], the pair of images shows images captured without HDR tone mapping (left) and with (right). Each pixel automatically determines the suitable integration time based on neighbouring pixel values by controlling the exposure time of individual pixels. Images are from [Martel et al., 2017, Martel et al., 2016].

1.6.3 Focal-plane Sensor-processor

The most widespread image formation on an imaging sensor (also referred to as the focal-plane) has two phases: exposure and readout. During the exposure, photons are captured by the photodiodes in each pixel, and then the stored capacity is read. The readout system is composed of one or several signal amplifiers and Analog-to-Digital Converters (ADCs), and the digitised output is then transferred to a host processor via a bus system for further processing.

The frame rate and latency of a real-time image processing pipeline are bounded by the maximum speed of image formation (from the photon to the digital signal latency).

The maximum frame rate of an imaging system depends on the following factors:

1. Exposure time required by the imager, which is dependent on the dynamic range of the camera and the lighting condition.
2. The total amount of pixels readout and the speed of the readout system.
3. The total volume of the digitised data and the data transfer rate of the interface.

The bottleneck for a fast and low-power image processing pipeline is the readout of a large number of pixels and its transfer to the processing unit. A significant slowdown in the frame rate occurs at the readout system when the electric charges, generated by the photons, are converted to digital

values since all pixels should go through the readout system [El-Desouki et al., 2009]. In fact, the readout system consumes 50% - 70% of the overall energy of the sensor [Likamwa et al., 2016].

Focal-Plane Sensor-Processor (FPSP) is a general-purpose vision chip technology that allows user-defined computation in a highly parallel manner on the focal-plane of the sensor at high frame rates [Zarándy, 2011]. The low energy, high frame rate nature of the FPSP – consuming only 1.23W even when operating at its maximum effective frame rate of 100,000 FPS [Carey et al., 2013b] – makes the device appealing for high speed and always-on applications. The key to the efficiency of FPSP is the ability to reduce the amount of data transferred. Compared to traditional camera sensors, FPSPs perform image processing at the earliest stage of the pipeline, on the focal-plane where the images are captured. The processed data is often more compact than the original image, and only the compressed data is transferred to the later stages, reducing bandwidth and energy consumption.

We focus on SCAMP-5 [Carey et al., 2013b], a 256×256 FPSP totaling 65,536 pixels. Each pixel combines a photodiode with a Processing Element (PE). PEs can execute an instruction simultaneously on their local data, resulting in Single Instruction Multiple Data (SIMD) parallel processing.

Each PE can store local data using 7 analog and 13 1-bit registers and perform simple computations such as logical and arithmetic operations. The arithmetic operations are carried out in the analog domain directly on the analog registers, eliminating the need for digitisation [Carey et al., 2013b]. Processing in analog is more efficient; however, this, in turn, introduces limitations [Carey et al., 2013a]. For instance, arithmetic operations become noisy. Moreover, analog values stored on the registers degrade gradually. After computation, data can be read out in different forms such as coordinates, binary frames, analog frames, or global data (*e.g.* regional summation) [Dudek and Hicks, 2005]. The device supports event readout for coordinate readout, where the cost is proportional to the number of events rather than the image dimension.

The small size of the sensor-processor chip limits the resources available on each PE. Such limitation can be addressed by registers shared among the pixel; however, at the cost of reducing the resolution of the sensor [Martel et al., 2015]. The instruction set is limited; only operations such as addition and subtraction are available, and for example, there is no multiplication. There is also no central memory; the PEs can only communicate data with their immediate adjacent pixels. While pixels far apart can communicate by iteratively passing on the messages, noise degrades the data as it is copied from one pixel to another. Despite all these constraints, several computer vision algorithms have been implemented on SCAMP-5 FPSP. Examples include: FAST keypoint detection [Chen et al., 2017], 4 DoF visual odometry [McConville et al., 2020], [Bose et al., 2017], [Debrunner et al., 2019], localisation [Castillo-Elizalde et al., 2021], target tracking [Greatwood et al., 2017], [Liu et al., 2021], depth estimation [Martel et al., 2017] and HDR-tone mapping [Martel et al., 2016] (as shown in Figure 1.2).

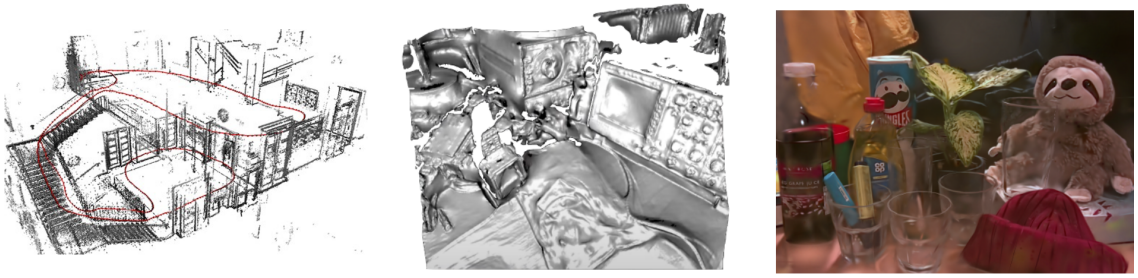


Figure 1.3: Comparison of different SLAM representations. **Left:** DSO [Engel et al., 2017], **Middle:** DTAM [Newcombe et al., 2011b], **Right:** Gaussian Splatting SLAM [Matsuki et al., 2024]. Images are adapted from the corresponding publications.

1.7 3D Scene Representation

Image data is large and contains many redundancies. As previously discussed, capturing images is energy inefficient and challenging to store and process if they are recorded over a long period. Visual SLAM provides a partial solution to this problem, whereby processing the continuous stream of images captured by a camera, SLAM estimates a 3D map and camera poses at every frame. This essentially is a form of *compression* where the input images are reduced into only the useful spatial information that can be used for downstream applications. However, as shown in the left and middle of Figure 1.3, most SLAM systems do not target photorealistic reconstruction; hence, they lose the details of the natural images. Recent SLAM systems aim to reconstruct the scene with photorealism [Rosinol et al., 2023, Sandström et al., 2023], including our system (right Figure 1.3) which we will discuss in Chapter 4.

Sparse SLAM methods focus on pose estimation [Mur-Artal et al., 2015, Engel et al., 2017, Forster et al., 2014] and the reconstructed 3D map is primarily useful for only localisation. Dense SLAM approaches often use voxel grids [Newcombe et al., 2011a, Dai et al., 2017, Whelan et al., 2015a, Prisacariu et al., 2014] or points [Keller et al., 2013, Whelan et al., 2015b, Schöps et al., 2019] as the underlying 3D representation to recover an interactable map, where the map can be used for downstream applications such as AR and robot navigation. While voxels enable a fast look-up of features in 3D, the representation is expensive, and the fixed voxel resolution and distribution are problematic when the spatial characteristics of the environment are not known in advance. On the other hand, a point-based map representation, such as surfel clouds, enables adaptive changes in resolution and spatial distribution by dynamically allocating point primitives in 3D space. While these dense approaches capture the scene's geometry well, they fail to capture the high-fidelity of natural scenes as the primitives are uncorrelated and not jointly optimised.

To capture a scene with photorealistic quality, differentiable volumetric rendering [Niemeyer et al., 2020] has recently been popularised with Neural Radiance Fields (NeRF) [Mildenhall et al., 2020]. Using a single Multi-Layer Perceptron (MLP) as a scene representation, NeRF performs volume rendering by marching along pixel rays, querying the MLP for opacity and colour. Since volume rendering is differentiable, the MLP representation is optimised to minimise the rendering loss

using multiview information to achieve high-quality novel view synthesis. The main weakness of NeRF is its training speed. Recent developments have introduced explicit volume structures such as multi-resolution voxel grids [Fridovich-Keil et al., 2022, Sun et al., 2022, Liu et al., 2020] or hash functions [Müller et al., 2022] to improve performance. Interestingly, these works demonstrate that the main contributor to high-quality novel view synthesis is not the neural network but rather differentiable volumetric rendering and that it is possible to avoid the use of an MLP and yet achieve comparable rendering quality to NeRF [Fridovich-Keil et al., 2022]. However, even with the incorporation of such explicit structures, per-pixel ray marching remains a significant bottleneck for rendering speed.

In contrast to NeRF, 3D Gaussian Splatting (3DGS) performs differentiable rasterisation. In 3DGS, a scene is represented by a large number of Gaussian blobs with orientation, elongation, colour and opacity, and similar to graphics rasterisations, by iterating over the primitives to be rasterised rather than marching along rays, 3DGS leverages the natural sparsity of a 3D scene and achieves an expressive representation that captures high-fidelity 3D scenes while offering significantly faster rendering. Several works have applied 3D Gaussians and differentiable rendering to static scene capture [Keselman and Hebert, 2022, Wang et al., 2022a], and in particular more recent works utilise 3DGS and demonstrate superior results in vision tasks such as dynamic scene capture [Luiten et al., 2024, Yang et al., 2024, Wu et al., 2024] and 3D generation [Tang et al., 2024, Yi et al., 2024].

Since 3D representations such as 3DGS can now efficiently capture high-fidelity scenes, they can compress all the input images into a coherent 3D scene with near-photorealistic rendering from any viewpoint. Hence, unlike the previous sparse/dense representation used in SLAM, photometric information is retained at near-original quality. The reconstructed representation is more lightweight than the original images combined, and the synthesised images can be used as input to a neural network such as segment anything model [Kirillov et al., 2023], Efficient SAM [Xiong et al., 2023] and surface normal estimation [Bae and Davison, 2024] as shown in Figure 1.4, which cannot be done with previous SLAM representations.

Having motivated Distributed Spatial AI from the perspective of near-sensor processing, scene representation, and decentralised algorithm, we now detail the contributions in this thesis.

1.8 Contributions

This thesis covers the following works:

- Riku Murai, Sajad Saeedi, Paul H.J. Kelly, **BIT-VO: Visual Odometry at 300 FPS using Binary Features from the Focal Plane**, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020 [Murai et al., 2020]. An extended version is published as:
  High-frame-rate Homography and Visual Odometry by Tracking Binary Features

Figure 1.4: Running different off-the-shelf pre-trained models on a novel-view synthesis from Gaussian Splatting SLAM [Matsuki et al., 2024]. **Top Row:** Self-captured monocular reconstruction is segmented using Efficient SAM [Xiong et al., 2023], **Bottom Row:** Reconstruction of Replica RGB-D dataset is converted into a surface normal estimate using DSINE [Bae and Davison, 2024].

---

from the Focal Plane, Autonomous Robots, 2023 [Murai et al., 2023b].

- Hidenobu Matsuki*, Riku Murai*, Paul H.J. Kelly, Andrew J. Davison, **Gaussian Splatting SLAM**, IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR), 2024 (*Equal contribution) [Matsuki et al., 2024]

- Riku Murai, Joseph Ortiz, Sajad Saeedi, Paul H.J. Kelly, Andrew J. Davison, **A Robot Web for Distributed Many-Device Localisation**, IEEE Transactions on Robotics (T-RO), 2023 [Murai et al., 2023a].

- Riku Murai, Ignacio Alzugaray, Paul H.J. Kelly, Andrew J. Davison, **Distributed Simultaneous Localisation and Auto-Calibration using Gaussian Belief Propagation**, IEEE Robotics and Automation Letters (RA-L), 2024 [Murai et al., 2024].

While not explicitly included in the thesis, the following works were done in conjunction:

- Matthew Lisondra, Junseo Kim, Riku Murai, Kourosh Zareinia, and Sajad Saeedi, **Visual Inertial Odometry using Focal Plane Binary Features (BIT-VIO)**, IEEE International Conference on Robotics and Automation (ICRA), 2024 [Lisondra et al., 2024]

- Aalok Patwardhan, Riku Murai, Andrew J. Davison, **Distributing Collaborative Multi-Robot Planning with Gaussian Belief Propagation**, IEEE Robotics and Automation Letters (RA-L), 2022 [Patwardhan et al., 2023]

- Alexandru-Iosif Toma, Hao-Ya Hsueh, Hussein Ali Jaafar, Riku Murai, Paul H.J. Kelly, Sajad Saeedi, **PathBench: A Benchmarking Platform for Classical and Learned Path Planning Algorithms**, 18th Conference on Robots and Vision (CRV), 2021 [Toma et al., 2021] An extended version is published as:
  Hao-Ya Hsueh, Alexandru-Iosif Toma, Hussein Ali Jaafar, Edward Stow, Riku Murai, Paul H.J. Kelly, Sajad Saeedi, **Systematic comparison of path planning algorithms using PathBench**, Advanced Robotics, 2022 [Hsueh et al., 2022]

- Edward Stow, Riku Murai, Sajad Saeedi, Paul H.J. Kelly, **Cain: Automatic Code Generation for Simultaneous Convolutional Filters on Focal-plane Sensor-processors**, Languages and Compilers for Parallel Computing (LCPC), 2020, [Stow et al., 2022b]. An extended version is published as:
  Edward Stow, Abrar Ahsan, Yingying Li, Ali Babaei, Riku Murai, Sajad Saeedi, Paul H.J. Kelly, **Compiling CNNs with Cain: focal-plane processing for robot navigation**, Autonomous Robots, 2022 [Stow et al., 2022a]

1.9 Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2 – Preliminaries. We formally introduce the probabilistic inference methods and graphical models, and then proceed to derive 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 – BIT-VO: Visual Odometry at 300 FPS using Binary Features from the Focal Plane. This chapter presents the work from our paper [Murai et al., 2020], which utilises SCAMP-5 FPSP for visual odometry. By performing image processing even before analogue-to-digital conversion, our system can operate in real-time at 300 FPS and demonstrate robustness against rapid, agile motions.

Chapter 4 – Gaussian Splatting SLAM. This chapter presents the work from our paper [Matsuki et al., 2024], which was carried out in close collaboration with Hidenobu Matsuki. We present the first application of 3D Gaussian Splatting in monocular SLAM and demonstrate a near-photorealistic 3D scene reconstruction, which can be used for many downstream applications (*e.g.* Figure 1.4).

Chapter 5 – Gaussian Belief Propagation For State Estimation. In this chapter, we derive Gaussian Belief Propagation (GBP) and extend it to support non-linearity and robust factors re-

quired for state estimation in robotics and computer vision. Additionally, we introduce and discuss how GBP can be extended to handle Lie group variables, which are required for localisation tasks.

Chapter 6 – A Robot Web for Distributed Many-Device Localisation. This chapter presents the work from our paper [Murai et al., 2023a], which performs decentralised multi-device localisation using GBP. Utilising advantageous properties of GBP, such as asynchronicity, we localise 1000s of robots using only ad-hoc peer-to-peer communication.

Chapter 7 – Distributed Simultaneous Localisation and Auto-Calibration using Gaussian Belief Propagation. This chapter presents the work from our paper [Murai et al., 2024], which extends on Robot Web to perform autocalibration of the sensors' and markers' extrinsic while simultaneously performing localisation and introduce adaptive regularisation to stabilise GBPs convergence when $\mathbf{SE}(3)$ variables are used.

Chapter 8 – Conclusion and Future Directions. Here, we conclude our thesis and discuss future research directions.

CHAPTER 2

Preliminaries

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 The Gaussian Distribution

The Gaussian distribution and its discovery are closely related to state estimation. Although initially discovered by Abraham de Moivre in 1733 as an approximation to a binomial distribution, Carl Friedrich Gauss rediscovered the Gaussian distribution when developing a *theory of errors* to model measurement uncertainty.

In 1801, an astronomer, Giuseppe Piazzi, observed an astronomical object, Ceres, which he believed to be a new, undiscovered planet (though now it's classified as a dwarf planet). However, only six weeks after the initial observation, Ceres disappeared behind the sun. Piazzi only made a few observations of Ceres in that period and could not accurately determine the orbit, making it challenging to estimate when and where it would reappear after passing behind the sun [Teets and Whitehead, 1999].

Many mathematicians and astronomers attempted to solve the problem; however, the mathematical tools used for astronomy at that time were insufficient to determine Ceres's celestial motion and led to many incorrect estimates. The relocalisation of Ceres was only made possible by Gauss, who, too, was working on the problem. To infer the orbit from very few noisy observations, he developed a *theory of errors* and used the least-squares method [1] to estimate the state which minimises the observation errors.

Gauss viewed the least-squares method probabilistically, where each observation x_i has a noise that follows some distribution $\phi(x_i)$. Gauss made the following assumption about $\phi(x_i)$:

1. Small errors are more likely than large errors.
2. The likelihood of errors is symmetric.
3. The best summarisation of multiple observations of the same quantity is the average.

Following these assumptions, for the solution of least-squares to be the maximum-likelihood estimator, he found and proved that the underlying noise must follow a bell-shaped distribution, a distribution we now call a Gaussian distribution [Stahl, 2006].

Since the discovery of the Gaussian distribution, its use in inference has been *ubiquitously successful* for over two centuries and has been applied to almost all parameter estimation prob-

[1] It is ambiguous whether Gauss used least-squares for localisation of Ceres, and Adrien-Marie Legendre was the first to publish least-squares method in 1806

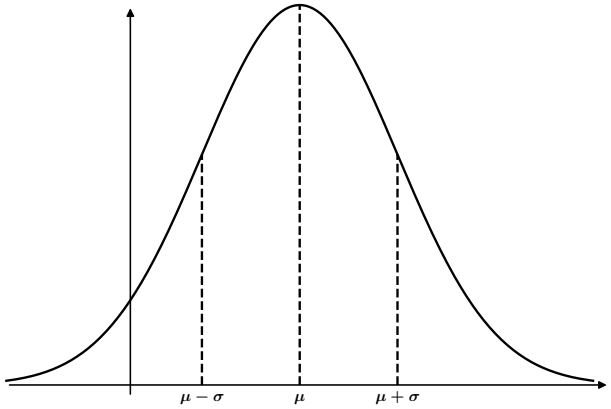


Figure 2.1: An example of a univariate Gaussian distribution, a symmetric bell-shaped curve.

lems [Jaynes, 2003]. Following the success, we will too use Gaussian distribution throughout the thesis. Hence, this section discusses the key properties of Gaussian distribution.

2.1.1 Univariate Gaussian

A univariate Gaussian distribution is defined as:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) , \tag{2.1}$$

where the distribution is fully characterised by the mean μ and the variance σ^2. As shown in Figure 2.1, Gaussian distribution is a bell-shaped curve. Any perturbation ε from the mean μ in either direction assigns equal likelihood, meaning $\mathcal{N}(\mu + \varepsilon; \mu, \sigma^2) = \mathcal{N}(\mu - \varepsilon; \mu, \sigma^2)$.

2.1.2 Multivariate Gaussian

Multivariate Gaussian is a generalisation of univariate Gaussian to a higher dimensional variable. A Gaussian with a d-dimensional state is defined as:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) , \tag{2.2}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean vector, $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ is the covariance matrix, and $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix.

The form of Equation 2.2 is called the **moment form** and alternative form parameterisation of multivariate Gaussian is called the **canonical form**:

$$\mathcal{N}^{-1}(\mathbf{x}; \boldsymbol{\eta}, \boldsymbol{\Lambda}) \propto \exp\left(-\frac{1}{2}\mathbf{x}^\top \boldsymbol{\Lambda}\mathbf{x} + \boldsymbol{\eta}^\top \mathbf{x}\right) . \tag{2.3}$$

$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix and $\boldsymbol{\eta} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ is the information vector. We can derive the canonical form from the moment from by:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{2.4}$$

$$= \exp\left(-\frac{1}{2}\mathbf{x}^\top\boldsymbol{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right), \tag{2.5}$$

$$\propto \exp\left(-\frac{1}{2}\mathbf{x}^\top\boldsymbol{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right). \tag{2.6}$$

By substituting in the definition of precision matrix and information vector, we have the Equation 2.3, the canonical representation. One advantage of the canonical form is that a product (up to a scale) is simply a summation of the parameters:

$$\mathcal{N}^{-1}(\mathbf{x}_\alpha; \boldsymbol{\eta}_\alpha, \boldsymbol{\Lambda}_\alpha)\mathcal{N}^{-1}(\mathbf{x}_\beta; \boldsymbol{\eta}_\beta, \boldsymbol{\Lambda}_\beta) = c\mathcal{N}^{-1}(\mathbf{x}_\gamma; \boldsymbol{\eta}_\alpha + \boldsymbol{\eta}_\beta, \boldsymbol{\Lambda}_\alpha + \boldsymbol{\Lambda}_\beta), \tag{2.7}$$

where c is the scaling term [Deisenroth et al., 2020, Chapter 6]. In the moment form, the operation is computationally harder as it involves computing inverses:

$$\mathcal{N}(\mathbf{x}_\alpha; \boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha)\mathcal{N}(\mathbf{x}_\beta; \boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta) = c'\mathcal{N}(\mathbf{x}_\gamma; \boldsymbol{\mu}_\gamma, \boldsymbol{\Sigma}_\gamma), \tag{2.8}$$

where $\boldsymbol{\mu}_\gamma = \boldsymbol{\Sigma}_\gamma(\boldsymbol{\Sigma}_\alpha^{-1}\boldsymbol{\mu}_\alpha + \boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta)$ and $\boldsymbol{\Sigma}_\gamma = (\boldsymbol{\Sigma}_\alpha^{-1} + \boldsymbol{\Sigma}_\beta^{-1})^{-1}$.

2.1.3 Marginalisation and Conditioning

Important and useful properties of Gaussian are that both marginalisation and conditioning of Gaussians are Gaussians. Let a joint Gaussian distribution be expressed as:

$$p(\mathbf{x}) = \mathcal{N}\left(\begin{pmatrix}\mathbf{x}_\alpha\\ \mathbf{x}_\beta\end{pmatrix}; \begin{pmatrix}\boldsymbol{\mu}_\alpha\\ \boldsymbol{\mu}_\beta\end{pmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{\alpha\alpha} & \boldsymbol{\Sigma}_{\alpha\beta}\\ \boldsymbol{\Sigma}_{\beta\alpha} & \boldsymbol{\Sigma}_{\beta\beta}\end{bmatrix}\right), \tag{2.9}$$

$$= \mathcal{N}^{-1}\left(\begin{pmatrix}\mathbf{x}_\alpha\\ \mathbf{x}_\beta\end{pmatrix}; \begin{pmatrix}\boldsymbol{\eta}_\alpha\\ \boldsymbol{\eta}_\beta\end{pmatrix}, \begin{bmatrix}\boldsymbol{\Lambda}_{\alpha\alpha} & \boldsymbol{\Lambda}_{\alpha\beta}\\ \boldsymbol{\Lambda}_{\beta\alpha} & \boldsymbol{\Lambda}_{\beta\beta}\end{bmatrix}\right). \tag{2.10}$$

Marginalisation In the moment form, marginalisation $p(\mathbf{x}_\alpha) = \int p(\mathbf{x})d\mathbf{x}_\beta$ is simply:

$$p(\mathbf{x}_\alpha) = \mathcal{N}(\mathbf{x}_\alpha; \boldsymbol{\mu}_M, \boldsymbol{\Sigma}_M), \tag{2.11}$$

where $\boldsymbol{\mu}_M = \boldsymbol{\mu}_\alpha$ and $\boldsymbol{\Sigma}_M = \boldsymbol{\Sigma}_{\alpha\alpha}$, selecting the sub-block of the mean and the covariance of the variables which we wish to keep. In the canonical form, it is more involved as it takes a Schur complement over the kept variables:

$$p(\mathbf{x}_\alpha) = \mathcal{N}^{-1}(\mathbf{x}_\alpha; \boldsymbol{\eta}_M, \boldsymbol{\Lambda}_M) \tag{2.12}$$

where $\boldsymbol{\eta}_M = \boldsymbol{\eta}_\alpha - \boldsymbol{\Lambda}_{\alpha\beta}\boldsymbol{\Lambda}_{\beta\beta}^{-1}\boldsymbol{\eta}_\beta$ and $\boldsymbol{\Lambda}_M = \boldsymbol{\Lambda}_{\alpha\alpha} - \boldsymbol{\Lambda}_{\alpha\beta}\boldsymbol{\Lambda}_{\beta\beta}^{-1}\boldsymbol{\Lambda}_{\beta\alpha}$.

Conditioning On the other hand, conditioning $p(\mathbf{x}_\alpha|\mathbf{x}_\beta) = p(\mathbf{x}_\alpha, \mathbf{x}_\beta)/p(\mathbf{x}_\beta)$ in moment form is hard:

$$p(\mathbf{x}_\alpha|\mathbf{x}_\beta) = \mathcal{N}(\mathbf{x}_\alpha; \boldsymbol{\mu}_C, \boldsymbol{\Sigma}_C) \,, \tag{2.13}$$

where $\boldsymbol{\mu}_C = \boldsymbol{\mu}_\alpha + \boldsymbol{\Sigma}_{\alpha\beta}\boldsymbol{\Sigma}_{\beta\beta}^{-1}(\mathbf{x}_\beta - \boldsymbol{\mu}_\beta)$ and $\boldsymbol{\Sigma}_C = \boldsymbol{\Sigma}_{\alpha\alpha} - \boldsymbol{\Sigma}_{\alpha\beta}\boldsymbol{\Sigma}_{\beta\beta}^{-1}\boldsymbol{\Sigma}_{\beta\alpha}$, and conditioning canonical form Gaussian is easy:

$$p(\mathbf{x}_\alpha|\mathbf{x}_\beta) = \mathcal{N}^{-1}(\mathbf{x}_\alpha; \boldsymbol{\eta}_C, \boldsymbol{\Lambda}_C) \,, \tag{2.14}$$

where $\boldsymbol{\eta}_C = \boldsymbol{\eta}_\alpha - \boldsymbol{\Lambda}_{\alpha\beta}\mathbf{x}_\beta$ and $\boldsymbol{\Lambda}_C = \boldsymbol{\Lambda}_{\alpha\alpha}$.

For a more in-depth discussion about the two parameterisation and SLAM, we refer the reader to [Eustice et al., 2005].

In summary, not only does the Gaussian distribution have many successful applications in state estimation, but it also exhibits desirable computational properties such as:

- Simple analytical solutions for many operations.
- Closed under operations such as marginalisation, conditioning, and product (only to a scale).

2.2 Methods for MAP inference

The Maximum a Posteriori (MAP) inference finds the point-estimate of the configuration which maximises the posterior and under a Gaussian assumption, this is equivalent to solving least-squares / regularised least-squares.

Given a posterior distribution $p(\mathbf{x}|\mathbf{z})$, MAP inference is defined as:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) \,, \tag{2.15}$$

where $\mathbf{x}$ is the state and $\mathbf{z}$ is the observations. Since logarithmic is a monotonically increasing function, $\arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) = \arg\min_{\mathbf{x}} -\log p(\mathbf{x}|\mathbf{z})$. Taking the logarithmic can be more computationally stable (as we avoid computation of the exponential if our posterior is an exponential distribution and the products can be expressed in summation), and the addition of a negative sign is because numerical optimisers historically tend to minimise the objective function rather than to maximise [Deisenroth et al., 2020, Chapter 8].

As stated, MAP estimate can be found by minimising the negative-log posterior:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} -\log p(\mathbf{x}|\mathbf{z}) \,, \tag{2.16}$$

$$= \arg\min_{\mathbf{x}} -\log p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) \,, \tag{2.17}$$

$$= \arg\min_{\mathbf{x}} -\log p(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}) \,. \tag{2.18}$$

And if we assume Gaussianity, both the likelihood $p(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{x})$ are Gaussian hence:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} -\log p(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}) \,, \tag{2.19}$$

$$= \arg\min_{\mathbf{x}} -\log \mathcal{N}(\mathbf{z}; h(\mathbf{x}), \boldsymbol{\Sigma}_z) - \log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) \,. \tag{2.20}$$

where $h(\cdot)$ is the measurement prediction function. Using Equation 2.2, the moment representation of a Gaussian distribution, and cancelling out the exponential with the logarithmic operation, our minimisation problem becomes:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} (\mathbf{z} - h(\mathbf{x}))^\top \boldsymbol{\Sigma}_z^{-1} (\mathbf{z} - h(\mathbf{x})) + (\mathbf{x} - \boldsymbol{\mu}_p)^\top \boldsymbol{\Sigma}_p^{-1} (\mathbf{x} - \boldsymbol{\mu}_p) \,. \tag{2.21}$$

To bring the equation into a least-squares form, following [Dellaert, 2005], we take the square root of the inverse covariances:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\boldsymbol{\Sigma}_z^{-\frac{1}{2}} (\mathbf{z} - h(\mathbf{x}))\|_2^2 + \|\boldsymbol{\Sigma}_p^{-\frac{1}{2}} (\mathbf{x} - \boldsymbol{\mu}_p)\|_2^2 \,, \tag{2.22}$$

or such can be written using a Mahalanobis distance as a weighted least-squares:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{z} - h(\mathbf{x})\|_{\Sigma_z}^2 + \|\mathbf{x} - \boldsymbol{\mu}_p\|_{\boldsymbol{\Sigma}_p}^2 \,. \tag{2.23}$$

We see that the prior term acts as regularisation of the state $\mathbf{x}$. Now that we know that MAP inference is equivalent to solving the least-squares problem, we will look at the different approaches we can take to solve the least-squares problem.

2.2.1 Linear Least Squares

Given a system of linear equations of form:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \,, \tag{2.24}$$

we want to find $\mathbf{x}$ which satisfies this equality. Typically, we solve this by minimising the *least squares problem*, which is defined as:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \,. \tag{2.25}$$

Since extreme points in convex problem have zero gradient, by taking the derivative of Equation 2.25 and setting it to zero, we can see that the optimal configuration $\mathbf{x}^*$ is indeed the same for both Equation 2.24 and Equation 2.25:

$$\nabla \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = \mathbf{A}^\top (\mathbf{A}\mathbf{x} - \mathbf{b}) = 0 \,, \tag{2.26}$$

Furthermore, we notice that the solution can be obtained by solving:

$$\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b} \,, \tag{2.27}$$

which is known as the normal equation.

2.2.2 Gradient Descent

We define a general objective function to minimise:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}) \,, \tag{2.28}$$

to find the optimal configuration $\mathbf{x}^*$.

Iterative methods, including gradient descent, can be written as finding the perturbation $\delta\mathbf{x}$ to apply to the current state $\mathbf{x}_t$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \delta\mathbf{x} \,. \tag{2.29}$$

The iterative formulation allows the methods to solve non-linear problems, for example, non-linear least squares, where a direct solution cannot be found (unlike a least-squares problem, where solving a normal equation yields the solution directly). By iteratively computing the update $\delta\mathbf{x}$ by linearising the non-linear system around the current estimate $\mathbf{x}_t$, the process continues until termination criteria (*e.g.* $\delta\mathbf{x}$ is small, upper bound on the number of iterations) are reached.

Gradient descent is a **first-order method** that iteratively updates the state by taking a step in the steepest descent:

$$\delta\mathbf{x} = -\alpha \nabla E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} \,, \tag{2.30}$$

where α is the learning rate. The learning rate is a hyperparameter that trades off between convergence speed and stability. If the learning rate is too large, the update step may be too large and the algorithm may diverge. On the other hand, if the learning rate is too small, the algorithm may take too long to converge. Alternatively, a line search can be employed to find the optimal learning rate at each step; however, this requires evaluation of the objective function per search and can be expensive.

A stochastic variant of gradient descent is often used in the context of Deep Learning (DL). As only the first-order information is used, gradient descent can efficiently compute the gradient using back-propagation, and such efficiency is crucial to DL since the number of parameters may reach orders of millions or even billions. Vanilla gradient descent often requires careful tuning of the learning rate, which is often challenging to do; hence, adaptive methods such as Adam [Kingma and Ba, 2015] are commonly used, where the algorithm tunes the learning rate per variable using past statistics.

2.2.3 Newton's Method

Newton's method is a **second-order method** that approximates the objective function as a quadratic function (using second-order derivative) around the current estimate $\mathbf{x}_t$ [Bertsekas, 1997, Chapter 1]. First, we perform Taylor expansion of $E(\mathbf{x})$ around $\mathbf{x}_t$:

$$E(\mathbf{x}) \approx E(\mathbf{x}_t) + \nabla E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}^\top (\mathbf{x} - \mathbf{x}_t) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_t)^\top \nabla^2 E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} (\mathbf{x} - \mathbf{x}_t) \,, \tag{2.31}$$

$$E(\mathbf{x}) \approx E(\mathbf{x}_t) + \nabla E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}^\top \delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^\top \nabla^2 E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} \delta\mathbf{x} \,, \tag{2.32}$$

where $\nabla^2 E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}$ is the Hessian matrix of the objective function at $\mathbf{x}_t$. We seek the update step which takes us to the critical point of the $E(\cdot)$ by setting the first-order derivative of the quadratic function to zero:

$$\nabla E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} + \nabla^2 E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}\delta\mathbf{x} = 0 \,, \tag{2.33}$$

and the update step is:

$$\delta\mathbf{x} = -\nabla^2 E(\mathbf{x})|^{-1}_{\mathbf{x}=\mathbf{x}_t} \nabla E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} \,. \tag{2.34}$$

Unlike gradient descent, no learning rate is required as it computes the step size, assuming that the underlying objective function is quadratic. Hence, when the objective function is indeed quadratic, Newton's method converges in a single step. While Newton's method is powerful, the $E(\cdot)$ may not be twice differentiable, and even if it is, computation of the Hessian matrix and taking its inverse is computationally expensive, especially if the problem is large.

2.2.4 Gauss-Newton Algorithm

If we have prior knowledge of the structure of the objective function, we can find a lightweight approximation of Newton's method. Gauss-Newton algorithm is an approximation of Newton's method which is used to solve non-linear least-squares problems. Given a least-squares objective function, we can rewrite the objective as:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}) \,, \tag{2.35}$$

$$= \arg\min_{\mathbf{x}} \frac{1}{2}\|r(\mathbf{x})\|^2_{\boldsymbol{\Sigma}} \,, \tag{2.36}$$

$$= \arg\min_{\mathbf{x}} \frac{1}{2} r(\mathbf{x})^\top \boldsymbol{\Lambda} r(\mathbf{x}) \,, \tag{2.37}$$

where $r(\mathbf{x}) = \mathbf{z} - h(\mathbf{x})$ is the residual function and $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the corresponding precision matrix which weights the residuals. Let $\mathbf{J} = \nabla r(\mathbf{x})^\top$ be the Jacobian of the residual function, the first-order derivative of the objective function is:

$$\nabla E(\mathbf{x}) = \mathbf{J}^\top \boldsymbol{\Lambda} r(\mathbf{x}) \,, \tag{2.38}$$

and the second-order derivative is:

$$\nabla^2 E(\mathbf{x}) = \mathbf{H}^\top \boldsymbol{\Lambda} r(\mathbf{x}) + \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} \,, \tag{2.39}$$

where $\mathbf{H} = \nabla^2 r(\mathbf{x})$ is the Hessian matrix of the residual function. Assuming that the current estimate $\mathbf{x}_t$ is close to the true solution $\mathbf{x}^*$, then $r(\mathbf{x}_t) \approx 0$. Hence, the second-order derivative can be approximated as:

$$\nabla^2 E(\mathbf{x}) \approx \mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J} \,. \tag{2.40}$$

Substituting back to Equation 2.34, the update step for Newton's method, we derive the update step for the Gauss-Newton method which is:

$$\delta\mathbf{x} = -(\mathbf{J}^\top \boldsymbol{\Lambda} \mathbf{J})^{-1} \mathbf{J}^\top \boldsymbol{\Lambda} r(\mathbf{x}) \,. \tag{2.41}$$

This is equivalent to solving the normal equation:

$$\mathbf{A}^\top \mathbf{A} \delta \mathbf{x} = \mathbf{A}^\top \mathbf{b} \tag{2.42}$$

where $\mathbf{A} = \mathbf{\Lambda}^{\frac{1}{2}} \mathbf{J}$ and $\mathbf{b} = \mathbf{\Lambda}^{\frac{1}{2}} r(\mathbf{x})$.

2.2.5 Levenberg-Marquardt Algorithm

The Gauss-Newton algorithms approximation does not hold if the initial estimate is far from the optimal solution. To avoid this problem, we can use the Levenberg-Marquardt algorithm (first discovered in [Levenberg, 1944] then rediscovered in [Marquardt, 1963]) which is a hybrid of the Gauss-Newton and the gradient descent algorithm. Intuitively, in the Levenberg-Marquardt algorithm, we take a Gauss-Newton step if we are close to the minimum, and otherwise, we take a gradient descent step. The factor λ interpolates the step between the Gauss-Newton step and the gradient descent step:

$$\delta \mathbf{x} = -(\mathbf{J}^\top \mathbf{\Lambda} \mathbf{J} + \lambda \mathbf{I})^{-1} \mathbf{J}^\top \mathbf{\Lambda} r(\mathbf{x}). \tag{2.43}$$

A small λ factor results in a Gauss-Newton step, and a large λ factor results in a gradient descent step. Marquardt's version takes the diagonal of the Hessian rather than using identity [Marquardt, 1963]:

$$\delta \mathbf{x} = -\left(\mathbf{J}^\top \mathbf{\Lambda} \mathbf{J} + \lambda \cdot \text{diag}(\mathbf{J}^\top \mathbf{\Lambda} \mathbf{J})\right)^{-1} \mathbf{J}^\top \mathbf{\Lambda} r(\mathbf{x}), \tag{2.44}$$

The λ factor is adaptively chosen where λ increases if the objective function increases, and λ decreases if the objective function decreases. When the objective function increases, the step is not taken and the choice of how much to increase or decrease the λ factor by is a hyperparameter [Gavin, 2019].

2.2.6 Exploiting Sparsity

For both the Gauss-Newton and Levenberg-Marquardt algorithms, we need to solve the normal equation to compute the update step. Whether we solve the normal equation using Cholesky or QR decomposition, the efficiency depends on the sparsity of the factorisation. Here, the elimination order of the variables (estimated state such as camera pose) is important to minimise the fill-ins (non-zero entries in the factorised matrix). Finding the optimal order of the variables to minimise the fill-ins is an NP-hard problem; however, there are many heuristics to reduce fill-ins, such as AMD [Amestoy et al., 1996], COLAMD [Davis et al., 2004] which are widely used in practice.

In some problems, exploiting the known structure of the problem enables efficient computation. For example, in bundle adjustment, camera poses and landmarks are ordered after each other such that the information matrix forms a block arrowhead matrix which can be efficiently solvable using Schur Complement Trick [Triggs et al., 1999].

Hence, the majority of the research in state estimation focuses on getting the structure and the ordering right; however, this requires knowledge of the global problem and is unsuitable for distributed computation. We imagine a network of many individual devices, and it is infeasible and

unrealistic to create a *global problem* which we can inspect and analyse. Instead, the devices must optimise locally and ideally achieve the same solution as their centralised counterparts.

2.2.7 Recovering Uncertainty using Laplace Approximation

Since MAP inference only yields a point-estimate, it is not a Bayesian procedure. However, we can approximate the distribution around the MAP solution, assuming that the underlying posterior distribution is a Gaussian. Such an approximation is often referred to as a Laplace approximation or Gaussian approximation [Murphy, 2012, Chapter 8] [Bishop, 2006, Chapter 4].

Let the posterior distribution $p(\mathbf{x}|\mathbf{z})$ be approximated as a Gaussian:

$$p(\mathbf{x}|\mathbf{z}) \approx \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto e^{-E(\mathbf{x})} , \tag{2.45}$$

where $E(\mathbf{x})$ is a negative log of the unnormalised posterior, often referred to as an **Energy Function**. The energy function around MAP solution $\mathbf{x}^*$ can be approximated using Taylor expansion:

$$E(\mathbf{x}) \approx E(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \nabla E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \nabla^2 E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x} - \mathbf{x}^*) . \tag{2.46}$$

We know that the MAP solution is at the critical point (where the gradient is 0), hence:

$$E(\mathbf{x}) \approx E(\mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \nabla^2 E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x} - \mathbf{x}^*) , \tag{2.47}$$

$$= E(\mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{H}(\mathbf{x} - \mathbf{x}^*) , \tag{2.48}$$

where $\mathbf{H} = \nabla^2 E(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}$ is the Hessian matrix.

Substituting the approximated energy function into Equation 2.45 yields us a Gaussian distribution:

$$p(\mathbf{x}|\mathbf{z}) \approx \frac{1}{Z} \exp\left(E(\mathbf{x}^*)\right) \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{H}(\mathbf{x} - \mathbf{x}^*)\right) \tag{2.49}$$

$$= \mathcal{N}(\mathbf{x}; \mathbf{x}^*, \mathbf{H}^{-1}) . \tag{2.50}$$

We can find the normalisation constant $Z = p(\mathbf{z})$ using the definition of multivariate Gaussian Equation 2.2:

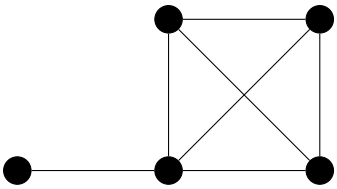
$$Z = p(\mathbf{z}) = \exp\left(E(\mathbf{x}^*)\right) \sqrt{(2\pi)^d |\mathbf{H}^{-1}|} , \tag{2.51}$$

where d is the dimension of $\mathbf{x}$.

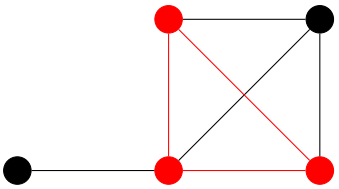
Equation 2.50 is referred to as the Laplace approximation to the posterior, and Equation 2.51 is the Laplace approximation to the marginal-likelihood. In the case of non-linear least-squares, as the estimate is at the true solution $\mathbf{x}^*$, we can approximate the Hessian using Equation 2.40.

2.3 Probabilistic Graphical Models

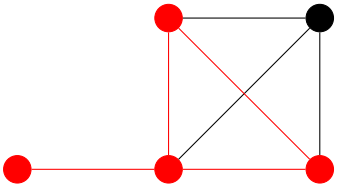
Real-world problems involve interactions amongst many variables, and probabilistic graphical models (PGMs) offer a visually intuitive method for modelling such problems. Additionally, they



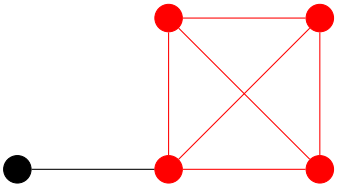
(a) A graph with vertices (black circles) and edges connecting them.



(b) An example of a subset of variables that forms a clique (highlighted in red).



(c) An example of a subset of variables does not form a clique (highlighted in red).



(d) An example of a subset of variables that forms a maximal clique.

Figure 2.2: Example of a clique and maximal clique.

reveal the sparsity of problems that enable efficient computation. In this section, we focus on factor graphs, a type of PGM which explicitly factorises a function into a set of factors.

2.3.1 Independence and Conditional Independence

First, we discuss the properties of independence and conditional independence that are essential for PGMs. Two variables $\mathbf{x}$ and $\mathbf{y}$ are independent if the joint distribution is equal to the product of the two marginals:

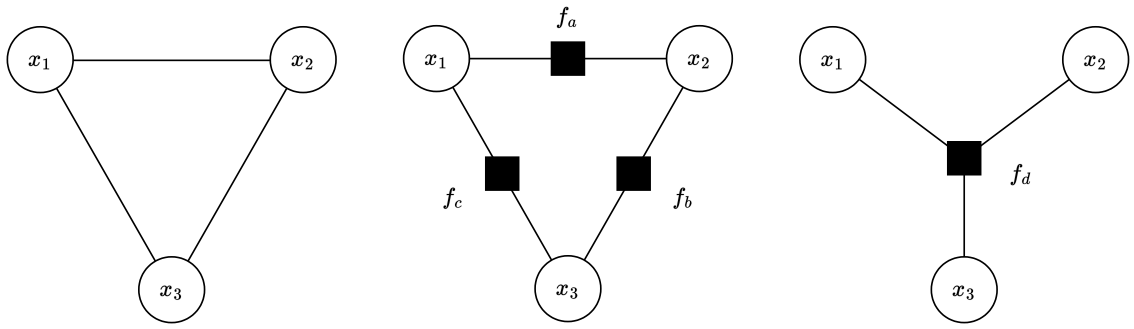
$$\mathbf{x} \perp\!\!\!\perp \mathbf{y} \iff p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}) \,. \tag{2.52}$$

This means that the variables cannot influence each other [Murphy, 2012, Chapter 2]. Finding independence simplifies the inference problem. However, we often don't observe such independence in real-world problems.

Problems, however, often have conditional independence, where the two variables are independent given knowledge of another variable. For example, $\mathbf{x}$ and $\mathbf{y}$ are conditionally independent given $\mathbf{z}$ if the joint distribution factorises into a product of two conditional distributions:

$$\mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z} \iff p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z}) \tag{2.53}$$

Conditional independence allows us to split a large probabilistic problem into small independent pieces that are easier to solve, and as such, finding and exploiting conditional independence is crucial for making inference tractable.



(a) $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ modelled using a UGM.

(b) $p_1(\mathbf{x})$ modelled using a factor graph.

(c) $p_2(\mathbf{x})$ modelled using a factor graph.

Figure 2.3: Example demonstrating how UGM loses the factorisation information.

2.3.2 Factor Graph

In an undirected graphical model (UGM), we can use the Hammersley-Clifford theorem to factorise a joint distribution $p(\mathbf{x})$ into:

$$p(\mathbf{x}) \propto \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c) \ , \tag{2.54}$$

where $\mathcal{C}$ is the set of maximal cliques in the graph, and $\mathbf{x}_c$ are the variables in the clique c. A clique is a subset of vertices where any two pairs of vertices in the subset are adjacent to each other. The maximal clique is a clique where no additional vertex can be added without violating the adjacency (Figure 2.2). ϕ_c is a potential/factor function assigned to each of the maximal cliques in the graph. They are a non-negative function of the arguments, and the joint distribution is proportional to the product of the potential function [Murphy, 2012, Chapter 19].

While UGM offers a method for representing factorisation, it cannot explicitly express the factors, and the underlying factorisation may get lost. For example, modelling the following using UGM will yield an identical graph:

$$p_1(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_3, x_1) \tag{2.55}$$

$$p_2(\mathbf{x}) = f_d(x_1, x_2, x_3) \tag{2.56}$$

For both cases, the UGM will be $p(\mathbf{x}) = \phi(x_1, x_2, x_3)$ as x_1, x_2, x_3 forms a clique as shown in Figure 2.3. Knowledge of the exact factorisation is crucial for efficient computation, and thus, we want to retain this information when we model the problem.

Instead of using UGM, we use a factor graph. A factor graph is an undirected bipartite graph [2] with two types of nodes: variable nodes $\mathbf{x} = \{x_i\}_{i=1:N_v}$ and factors nodes $\mathbf{f} = \{f_s\}_{s=1:N_f}$. The joint distribution $p(\mathbf{x})$ is factorised into:

$$p(\mathbf{x}) \propto \prod_{s=1}^{N_f} f_s(\mathbf{x}_s) \ , \tag{2.57}$$

[2] Bipartite graph is a graph with two disjoint and independent sets of vertices, and the edges only connect from one set to another.

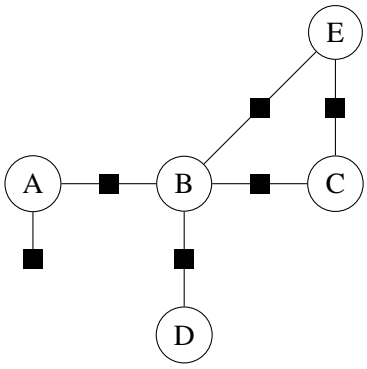


Figure 2.4: Example of a factor graph. The factor graph consists of variable nodes (circles) and factor nodes (squares).

where $\mathbf{x}_s = n(f_s)$ and $n(x)$, a neighbour function, returns the set of adjacent nodes directly connected to x. Each factor f_s is a function of a neighbouring set of variables $\mathbf{x}_s$, and unlike UGM, the factor does not correspond to a clique [Bishop, 2006, Chapter 8].

The graph explicitly represents the factorisation of the modelled joint distribution, and variables are conditionally independent given all other variables if they are not directly connected by an edge. Let x_i, x_j be variables which are not directly connected via an edge. They are conditionally independent given all other variables:

$$p(x_i, x_j|\mathbf{x}/\{x_i, x_j\}) = p(x_i|\mathbf{x}/\{x_i, x_j\})p(x_j|\mathbf{x}/\{x_i, x_j\}). \tag{2.58}$$

In the case of a factor graph, more specifically, two variables are conditionally independent if all paths between the nodes are blocked by the variable in the conditional. For example, in Figure 2.4, the variable A and C are conditionally independent given B, but B and C are not conditionally independent given E.

By modelling the posterior distribution using a factor graph, we can perform MAP inference using the algorithms we've discussed in Section 2.2. Once a MAP estimate is found, we can approximate the posterior distribution using a Laplace approximation.

2.4 Belief Propagation

Belief Propagation (BP) [Pearl, 1982] is an algorithm discovered in the 1980s to perform marginal inference on a factor graph. It is a Dynamic Programming (DP) algorithm where each node of the graph computes local quantities and passes the messages to its neighbours. The key advantage of BP is that it is a purely local algorithm, and each node independently can perform computation in parallel. Though originally BP was developed as an efficient method for exact marginal inference on a tree (graph which contains no cycles), the same method applied to an arbitrary graph is called Loopy Belief Propagation (LBP), which we will discuss in a later section.

In BP, as a factor graph is bipartite, variables pass messages to factors, and factors pass messages to variables. This message-passing process is iterative, where each node computes a new

outgoing message based on the latest messages it has received. After exchanging the messages, the variable's beliefs are updated based on the incoming messages. Each iteration of BP can be summarised in the following three steps:

- **Belief Update:** A variable's belief (the variable's marginal distribution) is computed by taking the product of all the latest incoming messages from the adjacent factors.
- **Variable-to-Factor Message-Passing:** The message from a variable to a factor represents the aggregate belief of the variable made by all other adjacent factors. This can be computed by downdating the variable's belief with the last message a receiving factor has sent.
- **Factor-to-Variable Message-Passing:** The message from a factor to a variable is the factor's belief of the variable, computed by marginalising all other variables adjacent to the factor.

While the order of operations does not matter and the message passing schedule can be arbitrary for the algorithm to converge, on a tree, belief converges to the exact marginal with one sweep of message passing from a root node to the leaf nodes and back.

BP and LBP are the key components of the thesis, and in this section, we will derive the BP algorithm to develop a better understanding of the algorithm. For clarity, we follow [Bishop, 2006, Chapter 8] and derive the rules for discrete variables; however, extension to continuous variables is simple, especially for the cases where we assume Gaussian distributions as we will show in Chapter 5.

Let $p(\mathbf{x})$ be a joint posterior represented by a factor graph containing no cycles, where $\mathbf{x}$ is the set of all variables. The joint posterior can be factorised into a product of factors using Equation 2.57. We are interested in computing the marginal distribution for each of the variables $\mathbf{x} = \{x_i\}_{i=1:N_v}$ which is achieved by summing over all other variables:

$$p(x_i) = \sum_{\mathbf{x}/x_i} p(\mathbf{x}) \, . \tag{2.59}$$

As a factor graph is bipartite, the variable x_i is connected to a neighbouring set of factors $n(x_i)$. As shown in Figure 2.5, each factor $f_\alpha \in n(x_i)$ branches off from x_i and forms a subtree F_α. The joint distribution of the subtree is $F_\alpha(x_i, \mathbf{X}_\alpha)$ where $\mathbf{X}_\alpha$ are the variables in the subtree α and note that since the overall graph is a tree, variables in each of the subtrees are disjoint.

With this factorisation, the joint distribution of the whole tree is:

$$p(\mathbf{x}) = \prod_{\alpha \in n(x_i)} F_\alpha(x_i, \mathbf{X}_\alpha) \, . \tag{2.60}$$

The marginal distribution $p(x_i)$ is computed by marginalising out all the variables apart from x_i:

$$p(x_i) = \sum_{\mathbf{x}/x_i} \prod_{\alpha \in n(x_i)} F_\alpha(x_i, \mathbf{X}_\alpha) \, . \tag{2.61}$$

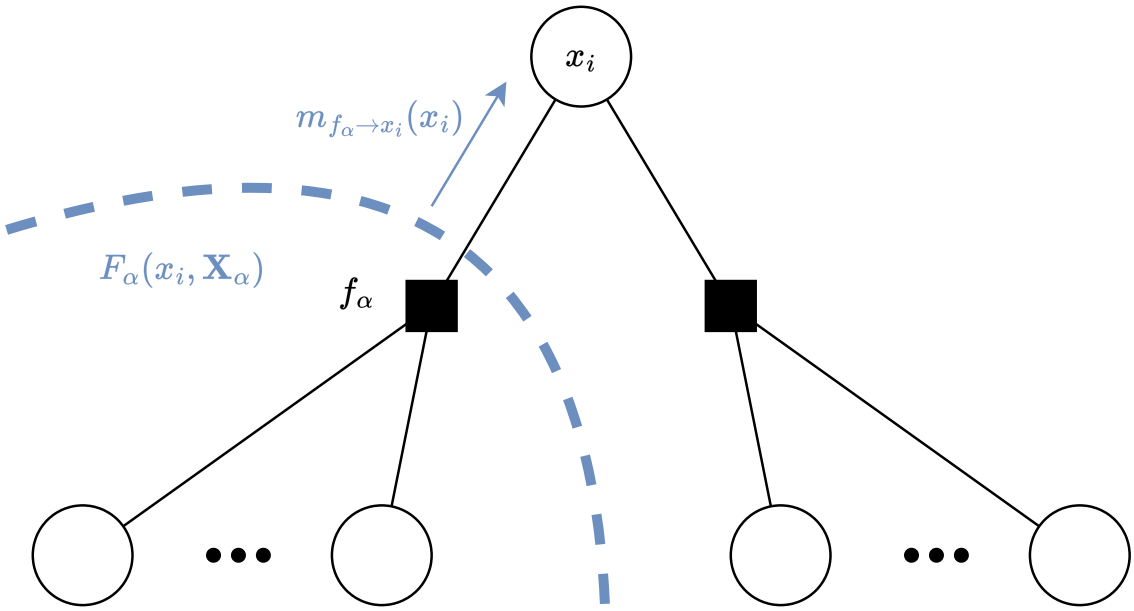


Figure 2.5: A variable node x_i connected to a factor f_α receives a message $m_{f_\alpha \to x_i}(x_i)$.

Since the set of variables $\{\mathbf{X}_\alpha : \alpha \in n(x_i)\}$ are disjoint, we can push in the summation inside the product of the factors:

$$p(x_i) = \prod_{\alpha \in n(x_i)} \sum_{\mathbf{X}_\alpha} F_\alpha(x_i, \mathbf{X}_\alpha) \ . \tag{2.62}$$

This interchange of summation and product is important. In Equation 2.61, we reconstruct the joint distribution of the whole tree (hence all variables) and then marginalise. On the other hand, in Equation 2.62, we first marginalise each subtree to create a function only of x_i. We then take the product of these functions to obtain the marginal distribution over x_i. Not only that, the latter is more efficient as we marginalise only the subtree. Such formulation enables a recursive definition of the algorithm, making the algorithm iterative.

Let $m_{f_\alpha \to x_i}(x_i)$ be the message from the factor f_α to the variable x_i. The message is the subtree's marginal belief of x_i:

$$m_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{X}_\alpha} F_\alpha(x_i, \mathbf{X}_\alpha) \ , \tag{2.63}$$

and Equation 2.61, the belief of variable x_i, can be written simply as the product of all incoming messages:

$$p(x_i) = \prod_{\alpha \in n(x_i)} m_{f_\alpha \to x_i}(x_i) \ . \tag{2.64}$$

This Equation 2.64 is the rule for **belief update**, where we combine the incoming messages from factors to the variable to obtain the marginal distribution of the variable x_i.

Now, we will keep unfolding our message-passing iterations to derive the variable-to-factor and factor-to-variable messages. Consider Figure 2.6 which focuses on the factor f_α, the subtree F_α is the product of f_α and all other factors in the subtree:

$$F_\alpha(x_i, \mathbf{X}_\alpha) = f_\alpha(\mathbf{x}_\alpha) \prod_{x_j \in \mathbf{x}_\alpha / x_i} G_j(x_j, \mathbf{X}_j) \ , \tag{2.65}$$

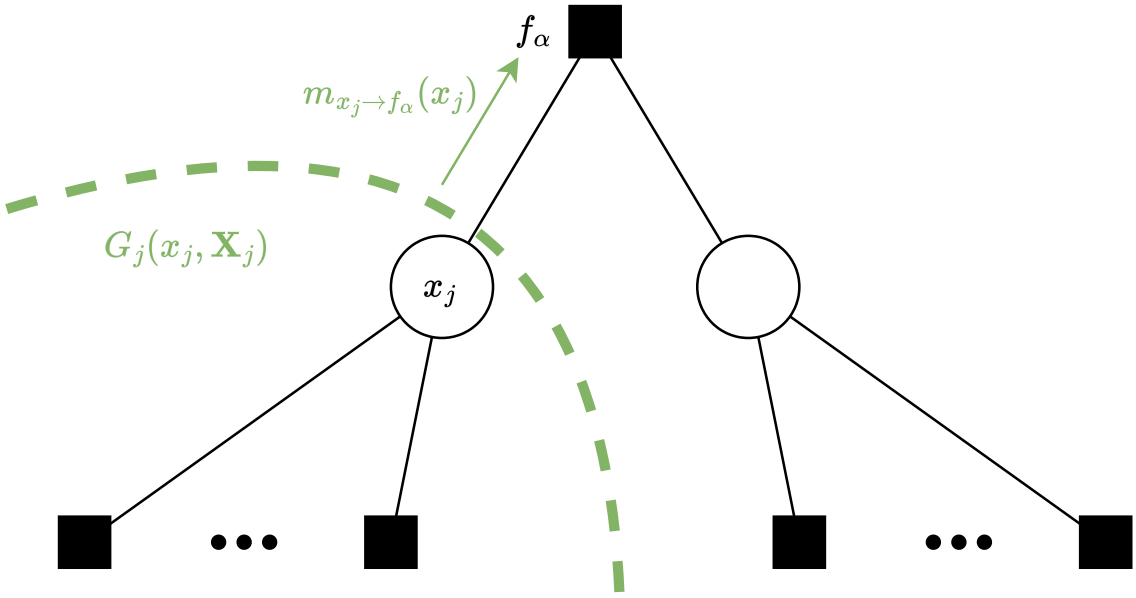


Figure 2.6: A factor node f_α connected to a variable x_j receives a message $m_{x_j \to f_\alpha}(x_j)$.

where $\mathbf{x}_\alpha = n(f_\alpha)$ is the set of parameters connected to the factor f_α, G_j is the joint distribution of the subtree stemming off variable $x_j \in \mathbf{x}_\alpha$, and $\mathbf{X}_j$ is the set of variables in the subtree G_j apart from variable x_j. Substituting back into Equation 2.63:

$$m_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{X}_\alpha} f_\alpha(\mathbf{x}_\alpha) \prod_{x_j \in \mathbf{x}_\alpha / x_i} G_j(x_j, \mathbf{X}_j) \,. \tag{2.66}$$

We can separate $\mathbf{X}_\alpha$, all the variables in the subtree F_α, into $\mathbf{x}_\alpha / x_i$ and $\mathbf{X}_j$. The variable x_j is not part of subtree F_α hence is. Again, $\mathbf{x}_\alpha$ and $\mathbf{X}_j$ are disjoint as the graph is a tree. This allows us to rewrite the message as:

$$m_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{x}_\alpha / x_i} \sum_{\mathbf{X}_j} f_\alpha(\mathbf{x}_\alpha) \prod_{x_j \in \mathbf{x}_\alpha / x_i} G_j(x_j, \mathbf{X}_j) \,. \tag{2.67}$$

Using the same reordering trick as Equation 2.61, Equation 2.62, we can push the summation inside the product:

$$m_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{x}_\alpha / x_i} f_\alpha(\mathbf{x}_\alpha) \prod_{x_j \in \mathbf{x}_\alpha / x_i} \sum_{\mathbf{X}_j} G_j(x_j, \mathbf{X}_j) \,. \tag{2.68}$$

and by defining $m_{x_j \to f_\alpha}(x_j)$ as the message from the variable x_j to the factor f_α:

$$m_{x_j \to f_\alpha}(x_j) = \sum_{\mathbf{X}_j} G_j(x_j, \mathbf{X}_j) \,, \tag{2.69}$$

we can write the message from the factor f_s to the variable x_i as:

$$m_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{x}_\alpha / x_i} f_\alpha(\mathbf{x}_\alpha) \prod_{x_j \in \mathbf{x}_\alpha / x_i} m_{x_j \to f_\alpha}(x_j) \,, \tag{2.70}$$

the recursive formulation of the **factor-to-variable message**. To find the message to x_i, we marginalise out all the variables connected to the factor apart from x_i from subtree F_α, which is a joint distribution formed via a product of the factor f_α and subtrees $G_j \in n(f_\alpha)/x_i$.

Finally, we expand Equation 2.69 to find the definition of variable-to-factor message. G_j is the joint distribution of a subtree with x_j as the root, and this can be factorised into:

$$G_j(x_j, \mathbf{X}_j) = \prod_{\beta \in n(x_j)/\alpha} F_\beta(x_j, \mathbf{X}_\beta) \ . \tag{2.71}$$

Substituting this into Equation 2.69, the definition of the message from the variable x_j to the factor f_α:

$$m_{x_j \to f_\alpha}(x_j) = \sum_{\mathbf{X}_j} \prod_{\beta \in n(x_j)/\alpha} F_\beta(x_j, \mathbf{X}_\beta) \ , \tag{2.72}$$

and again using the reordering trick to push the summation inside:

$$m_{x_j \to f_\alpha}(x_j) = \prod_{\beta \in n(x_j)/\alpha} \sum_{\mathbf{X}_\beta} F_\beta(x_j, \mathbf{X}_\beta) \ . \tag{2.73}$$

Using the definition of factor-to-variable message, Equation 2.63, we can rewrite the subtree's F_β marginal belief of x_j as a message $m_{f_\beta \to x_j}(x_j)$. This allows us to write the **variable-to-factor message** as:

$$m_{x_j \to f_\alpha}(x_j) = \prod_{\beta \in n(x_j)/\alpha} m_{f_\beta \to x_j}(x_j) \ . \tag{2.74}$$

To perform inference using the BP algorithm, from a tree, we arbitrarily choose a root node x_i. The marginal distributions of all the variables in the tree are computed by making a sweep of messages along the tree, passing the messages from the leaves to the root, and then from the root to the leaves. The recursive message passing is initialised by setting the messages from the leaf nodes to be $m_{f_s \to x_j}(x_j) = f_s(x_j)$ if the leaf is a factor or $m_{x_j \to f_s}(x_j) = 1$ if the leaf is variable.

Summary: The marginal distribution of a variable x_i is the product of all incoming messages:

$$p(x_i) = \prod_{\alpha \in n(x_i)} m_{f_\alpha \to x_i}(x_i) \ . \tag{2.75}$$

and the message from the factor f_α to the variable x_i is:

$$m_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{x}_\alpha / x_i} f_\alpha(\mathbf{x}_\alpha) \prod_{x_j \in \mathbf{x}_\alpha / x_i} m_{x_j \to f_\alpha}(x_j) \ . \tag{2.76}$$

where the message from the variable x_i to the factor f_α is:

$$m_{x_i \to f_\alpha}(x_i) = \prod_{\beta \in n(x_i)/\alpha} m_{f_\beta \to x_i}(x_i) \ . \tag{2.77}$$

2.4.1 Loopy Belief Propagation

So far, we have assumed that the factor graph is a tree, and BP is guaranteed to converge to the exact solution in such cases. However, most practical and interesting problems involve cycles in the graph, which violates the independence assumption of the subtrees which we've made in BP derivation.

Although BP was not designed for non-tree graphs, as the formulation of BP is iterative, one can directly apply the algorithm on an arbitrary graph and keep iterating until the variables converge. Such an approach is called LBP and initially, it was simply a heuristic to perform inference on an arbitrary graph and lacked theoretical understanding. However, empirically, LBP performed well on an arbitrary graph with cycles and often converged to a sensible solution [Murphy et al., 1999]. Later, a mathematical grounding was provided, where LBP is actually equivalent to performing a variational inference, minimising Bethe Free energy [Yedidia et al., 2001]. Under Gaussian assumption, when LBP converges, the means are exact, however, the covariances are not and are often overconfident [Weiss and Freeman, 1999].

2.5 State Estimation and Lie Groups

Up to this point, the underlying assumption has been that the state space is Euclidean. Robotics requires rotation as part of the states; however, a naive parameterisation of 3D rotation such as Euler angles suffers from Gimbal lock – a degenerate configuration in which the parameters can only rotated in two-dimensions – making it unsuitable for state estimation. In order to avoid such degeneracy, rotations are *over-parameterised*, with more parameters than the Degrees of Freedom (DoF). For instance, a rotation matrix, a 3×3 matrix, has 9 parameters (thereby spanning 9 dimensions) but possesses only 3 DoF. As more parameters are used than the underlying DoF in such parameterisation, not all 3×3 matrices are valid rotation matrices, and in the case of a rotation matrix, the matrix is constrained to be orthonormal and have unit determinant, and in the case of a quaternion, the vector must be unit-norm.

When optimising an over-parameterised representation, the dimension of the states is larger than the DoF. Performing gradient descent may result in a state which violates the imposed constraints and requires some projection to bring the solution back into a valid subspace. Such approaches are indeed feasible; however, we here take an elegant alternative and utilise Lie theory to optimise the rotations and rigid transformations. The use of Lie theory has multiple advantages:

- Commonly used states such as rotations $\mathbf{SO}(2), \mathbf{SO}(3)$ and rigid transformations $\mathbf{SE}(2), \mathbf{SE}(3)$ are Lie Groups, and this allows the formulation of a general optimisation framework which works across various states. For example, both non-linear least-squares and Extended-Kalman filter (EKF) that work in the Euclidean space require only a simple modification to support the Lie group. Once the Lie group is supported, the algorithm works across different states, regardless of whether it is 2D rotation or 3D rigid transformation.
- The internal details of the representation (i.e. for rotation, whether we use a rotation matrix or quaternion) can be abstracted away since quaternion manifold $\mathbf{S}^3$ is a double cover of $\mathbf{SO}(3)$ and isomorphic up to the first cover (*i.e.* if the scalar part of quaternion is greater than 0). An application may require or is more efficient with a different representation; however, the core algorithm (*e.g.* the non-linear least-squares framework) does not need to change if we abstract the details away using Lie theory.

- By formulating the optimisation problem using a minimal representation, we can avoid redundant computations. For example, the rotation matrix has 9 parameters, but only 3 DoF, so the Jacobian derived using minimal parameterisation is $\mathbb{R}^{m\times 3}$ not $\mathbb{R}^{m\times 9}$ (as we will show in Chapter 4 where we derive the camera pose Jacobian on Lie group).

Now that we've motivated why Lie theory is useful, in the next section, we will briefly introduce Lie theory which we will use in our thesis.

2.6 Lie Groups and Lie Algebras

Lie groups and Lie algebras, named after a Norwegian mathematician Marius Sophus Lie, are essential components of modern robotics and state estimation to formulate the problem correctly to obtain a precise and stable solution [Solà et al., 2018, Barfoot, 2017].

While the Lie theory is abstract and not simple, fortunately for us, only a part of the theory is required in estimation for robotics. We will only touch on the theory and keep this section as minimal as possible. We refer the reader to the excellent tutorial *"A micro Lie theory for state estimation in robotics"* [Solà et al., 2018] for further details.

2.6.1 Group Properties

A Lie group is, as the name suggests, a group, and a group G is defined to be a set with the following properties:

- **Closure**: If $\mathcal{X}, \mathcal{Y} \in G$ then the composition $\mathcal{X} \circ \mathcal{Y}$ is also in G.
- **Associativity**: The composition is associative, that is for $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in G$, $(\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z})$.
- **Identity**: There is an identify element I, where $I \circ \mathcal{X} = \mathcal{X} \circ I = \mathcal{X}$.
- **Inverse**: Each element has its inverse, such that $\mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{X} \circ \mathcal{X}^{-1} = I$.

Group properties provide an abstraction over the operations that we apply to the group elements, clarifying, for example, that the composition of elements of the group remains in the group.

2.6.2 Group Action

A Lie group element can act on a non-group element (*e.g.* $\mathbb{R}^n$). Formally, G acts on set X, if for all point $x \in X$, a function $\cdot : G \times X \to X$ which satisfies the following:

- Identity: $I \cdot x = x$

- Compatibility: $(\mathcal{X} \circ \mathcal{Y}) \cdot x = \mathcal{X} \cdot (\mathcal{Y} \cdot x)$.

Group actions are used to transform the non-group elements, performing rotations, translations, scalings and combinations of them. As a concrete example, let $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbf{SE}(3)$. The $\mathbf{SE}(3)$ group action on a point $\mathbf{x} \in \mathbb{R}^3$ is defined as: $\mathbf{T} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x} + \mathbf{t}$, a rigid transformation of a point in 3D space.

2.6.3 Manifolds

A Lie group is not only a group but is also a manifold. A d-dimensional manifold $\mathcal{M}$ is a topological space where around any point $\mathcal{X} \in \mathcal{M}$ there is a neighbourhood which is homeomorphic to $\mathbb{R}^d$ (i.e. the space is locally similar to Euclidean space). Such a neighbourhood forms a **tangent space** around $\mathcal{X}$, denoted as $T_{\mathcal{X}}\mathcal{M}$. Additionally, the Lie group is a smooth manifold; hence, the manifold is differentiable everywhere.

A vector defined in the tangent space is called a **tangent vector**. In the Lie groups case, they are Lie algebra $\mathfrak{m}$ which are isomorphic to $\mathbb{R}^M$, which means we can perform linear algebra operations on the tangent vector. Formally, $\mathfrak{m}$ and $\mathbb{R}^M$ are related to each other using Hat and Vee operators:

$$\text{Hat} : \mathbb{R}^M \to \mathfrak{m}; \quad \boldsymbol{\tau} \mapsto \boldsymbol{\tau}^\wedge = \sum_{i=1}^{M} \tau_i \mathbf{E}_i \ , \tag{2.78}$$

$$\text{Vee} : \mathfrak{m} \to \mathbb{R}^M; \quad (\boldsymbol{\tau}^\wedge)^\vee \mapsto \boldsymbol{\tau} = \sum_{i=1}^{M} \tau_i \mathbf{e}_i \ , \tag{2.79}$$

where $\mathbf{E}_i$ are the generators of $\mathfrak{m}$, and $\mathbf{e}_i$ are the basis of $\mathbb{R}^M$. They are related via the Hat operator: $\mathbf{E}_i = \mathbf{e}_i^\wedge$. As a concrete example, the Hat operator for $\mathfrak{so}(3)$ is a skew-symmetric matrix:

$$\text{Hat} : \mathbb{R}^3 \to \mathfrak{so}(3); \quad \boldsymbol{\tau} \mapsto \boldsymbol{\tau}^\wedge = \begin{bmatrix} 0 & -\tau_3 & \tau_2 \\ \tau_3 & 0 & -\tau_1 \\ -\tau_2 & \tau_1 & 0 \end{bmatrix} \tag{2.80}$$

$$= [\boldsymbol{\tau}]_\times \ . \tag{2.81}$$

Vee is the inverse, whereby we select the appropriate elements from the skew-symmetric matrix.

2.6.4 Exponential Map and Logarithmic Map

Every Lie group element is associated with a Lie algebra, and is related through exponential and logarithmic maps.

The exponential map on an arbitrary square matrix A is defined as:

$$\exp(\mathbf{A}) = \sum_{n=0}^{\infty} \frac{\mathbf{A}^n}{n!} \ . \tag{2.82}$$

Taking an exponential of Lie algebra is an exact mapping to an element in the Lie group, defined as, $\exp : \mathfrak{m} \to \mathcal{M}$. Luckily, for many commonly used Lie algebras / Lie groups, the exponential and logarithmic map has a closed-form solution. For example, in case of $\boldsymbol{\tau}^\wedge \in \mathfrak{so}(3)$, the exponential map is defined as:

$$\exp(\boldsymbol{\tau}^\wedge) = \mathbf{I} + \frac{\sin(\theta)}{\theta}[\boldsymbol{\tau}]_\times + \frac{1-\cos(\theta)}{\theta^2}[\boldsymbol{\tau}]_\times^2 , \tag{2.83}$$

where $\theta = \|\boldsymbol{\tau}\|_2$.

The logarithmic map is the inverse of the exponential map, $\log : \mathcal{M} \to \mathfrak{m}$, and hence:

$$\boldsymbol{\tau}^\wedge = \log\left(\exp(\boldsymbol{\tau}^\wedge)\right), \text{ and } \mathcal{X} = \exp\left(\log(\mathcal{X})\right) . \tag{2.84}$$

The logarithmic map can be found by inverting the exponential map. To simplify the notation, we define a direct mapping from $\mathbb{R}^N$ to the Lie group, and vice versa:

$$\text{Exp} : \mathbb{R}^M \to \mathcal{M}; \quad \boldsymbol{\tau} \mapsto \mathcal{X} = \exp(\boldsymbol{\tau}^\wedge) , \tag{2.85}$$

$$\text{Log} : \mathcal{M} \to \mathbb{R}^M; \quad \mathcal{X} \mapsto \boldsymbol{\tau} = \log(\mathcal{X})^\vee , \tag{2.86}$$

avoiding the use of Hat and Vee operators.

2.6.5 Moving on a Manifold

We define two operations on the manifold: retraction $\mathcal{R} : \mathcal{M} \times T_\mathcal{X}\mathcal{M} \to \mathcal{M}$, and its inverse $\mathcal{L} : \mathcal{M} \times \mathcal{M} \to T_\mathcal{X}\mathcal{M}$. We can view retraction as moving a point on a manifold by a vector and its inverse as finding a vector between two points on a manifold. While the retraction may not be globally invertible (i.e. multiple tangent vectors can retract to the same point on the manifold due to, for example, rotation wrap-around), locally, one can define one-to-one inversion. In the case of a Lie group, we use the exponential map and logarithmic map:

$$\mathcal{Y} = \mathcal{R}(\mathcal{X}, \boldsymbol{\tau}) \triangleq \mathcal{X} \oplus \boldsymbol{\tau} \triangleq \mathcal{X} \circ \text{Exp}(\boldsymbol{\tau}) \in \mathcal{M} , \tag{2.87}$$

$$\boldsymbol{\tau} = \mathcal{L}(\mathcal{X}, \mathcal{Y}) \triangleq \mathcal{Y} \ominus \mathcal{X} \triangleq \text{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \in T_\mathcal{X}\mathcal{M} . \tag{2.88}$$

Here, we've introduced the $\oplus$ and $\ominus$ operators, which are the shorthand notation.

Retraction and its inverse act as a useful tool for moving freely along the surface of the manifold. Such operations allow us to optimise states using iterative methods such as Gauss-Newton as if they were in Euclidean space and then retract them back onto the manifold.

When using Gauss-Newton in Euclidean space, we compute the increment $\delta\mathbf{x}$ to the state $\mathbf{x}$ which minimises the objective function $E(\mathbf{x})$ around the current linearisation point:

$$\delta\mathbf{x}^* = \underset{\delta\mathbf{x}}{\arg\min}\, E(\mathbf{x} + \delta\mathbf{x}) . \tag{2.89}$$

As long as we formulate the optimisation problem to compute the increment, the extension to the manifold is trivial. Given that the state $\mathbf{x}$ is a Lie Group, we can compute the increment $\delta\boldsymbol{\tau}$ in the tangent space, minimising the following:

$$\delta\boldsymbol{\tau}^* = \underset{\delta\boldsymbol{\tau}}{\arg\min}\, E(\mathbf{x} \oplus \delta\boldsymbol{\tau}) . \tag{2.90}$$

where $\delta\boldsymbol{\tau}$ is the increment in the tangent space. We update the state by retracting the increment back onto the manifold:

$$\mathbf{x}_{t+1} = \mathbf{x}_t \oplus \delta\boldsymbol{\tau}^* \,, \tag{2.91}$$

iteratively updating the state until convergence.

2.6.6 Lie Group Derivatives

Using the retraction and its inverse, we can define derivatives on functions that use Lie Group elements. First, the standard definition of derivatives is:

$$\frac{df(\mathbf{x})}{d\mathbf{x}} = \lim_{\mathbf{x}\to 0} \frac{f(\mathbf{x}+\delta\mathbf{x}) - f(\mathbf{x})}{\delta\mathbf{x}} \,, \tag{2.92}$$

where the function is multivariate $f : \mathbb{R}^m \to \mathbb{R}^n$, and the resulting derivative (Jacobian matrix) is a $n \times m$ matrix. Now, we will extend the standard definition of derivatives to a function which operates on the Lie group elements. By redefining the function $f : \mathcal{M} \to \mathcal{N}$, using $\{\oplus, \ominus\}$, we get:

$$\frac{\mathcal{D}f(\mathcal{X})}{\mathcal{D}\mathcal{X}} = \lim_{\boldsymbol{\tau}\to 0} \frac{f(\mathcal{X}\oplus\boldsymbol{\tau}) \ominus f(\mathcal{X})}{\boldsymbol{\tau}} \,, \tag{2.93}$$

where $\boldsymbol{\tau}$ is the tangent vector in the tangent space $T_{\mathcal{X}}\mathcal{M}$, we now express the derivative using infinitesimal deviation in the tangent space, and the dimension of the Jacobian matrix is $n \times m$ which corresponds to the degrees of freedom, not the dimension of $\mathcal{M}, \mathcal{N}$. We will use this to derive the camera pose Jacobian against 3D Gaussians in Chapter 4.

2.7 Summary

This section covered the preliminary materials required for the rest of the thesis, mainly covering probabilistic methods, Belief Propagation and Loopy BP, and an introduction to Lie theory. We utilise MAP point-estimation for Visual Odometry (VO) (Chapter 3) and Lie theory for Simultaneous Localisation and Mapping (SLAM) (Chapter 4) to derive the camera pose Jacobian against a 3D Gaussian map. BP and Lie theory are utilised in Chapter 5 to derive Gaussian Belief Propagation (GBP) which we then extend to support the Lie group variables.

CHAPTER 3

BIT-VO: Visual Odometry at 300 FPS using Binary Features from the Focal Plane

Contents

A Focal-Plane Sensor-Processor (FPSP) is a next-generation camera technology which enables every pixel on the sensor chip to perform computation in parallel on the focal-plane where the light intensity is captured. SCAMP-5 is a general-purpose, user-programmable FPSP used in this work, which performs computations in the *analog domain* before Analog-to-Digital Converter (ADC). By extracting visual features from an image on the focal-plane, we minimise the amount of digitised and transferred data, and as a consequence, SCAMP-5 offers a high frame rate operation while maintaining a low energy consumption. Here, we present BIT-VO, which is the first[1] 6-Degrees of Freedom Visual Odometry (VO) algorithm which utilises the FPSP. Our entire system operates at 300 FPS in a natural environment, using binary edges and corner features detected by the SCAMP-5.

3.1 Introduction

In this chapter, we explore an *unconventional* approach to the problem of low-power, high-frame rate vision. Instead of processing the captured image data on a centralised processing unit, we compute visual features *directly* on the sensor, performing early-vision computation even before the ADC.

For many reasons, vision-based pose-estimation algorithms such as VO and Visual Simultaneous Localisation and Mapping (SLAM) benefit from higher frame rates. First, the small inter-frame motion enables faster and more reliable camera pose tracking [Handa et al., 2012], and second, the reduced motion blur is beneficial for feature extraction. Typically, state-of-the-art odometry and SLAM systems operate at the frame rate of a conventional camera, which ranges from 30 to 60 FPS. While one can use a high-speed camera [Gemeiner et al., 2008], increasing the frame rate increases the energy consumption and the volume of data to be processed. Hence, operating at a higher frame rate demands more computation in a shorter time frame. In a sparse VO/SLAM system such as ORB-SLAM [Mur-Artal and Tardós, 2017], a sparse set of visual features are extracted from each image, and only such features are used for the rest of the SLAM pipeline. Unfortunately, feature extraction is often computationally expensive (11ms for 1000 ORB features) [Mur-Artal and Tardós, 2017], and prevents the system from running at a higher frame rate, even if we did have access to a high-speed camera. The bottleneck is that images are first transferred from a sensor to a processing unit, then the visual features are extracted. Instead, this chapter explores a different way and streams just the relevant features from the image sensor.

Using an FPSP sensor discussed in Section 1.6.3, as opposed to traditional camera sensors, we perform visual feature extraction on the sensor itself to deliver a reduced volume of data to later stages – in this chapter, just binarised corners and edges. Offloading the feature extraction to the FPSP reduces bandwidth, energy consumption and computational overhead on the host device.

[1] 6 DoF Visual Odometry using SCAMP-5 was attempted during a Master's project; however, it was limited to artificial and small scenes [Murai, 2019].

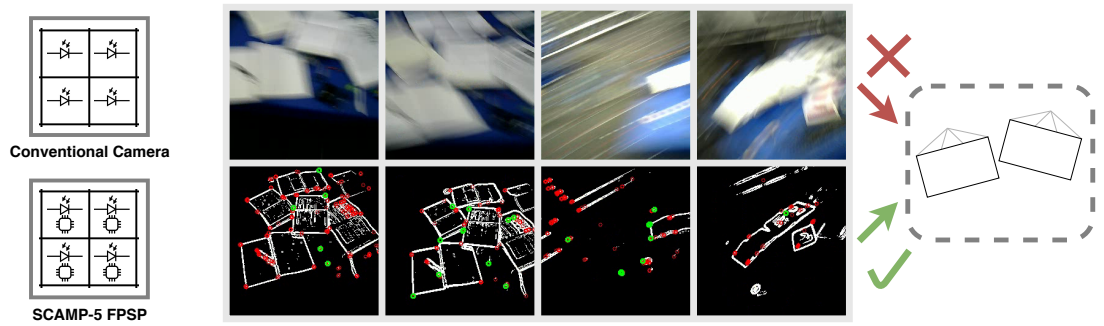


Figure 3.1: Comparison of the data used by our proposed VO vs conventional VOs. Our system does not use intensity images (top row) but uses the binary edges and corners (bottom row) extracted by SCAMP-5 at 300 FPS. Notice that the edges, when extracted at a high frame rate, are tolerant against motion blur and are sharp even when the device is subject to violent motions. For the conventional camera, such motion severely blurs the images.

Like FPSPs, event cameras are another low-power, low-latency camera technology, which outputs an asynchronous stream of intensity changes [Lichtsteiner et al., 2008]. Many VO/SLAM algorithms have been implemented using event cameras [Gallego et al., 2020] and we've summarised them in Section 1.6.1; however, the bandwidth of data transferred is proportional to the manoeuvre speed of the actual camera. On the other hand, an FPSP outputs data at a consistent rate; thus, there is no significant fluctuation in the amount of data transferred under any motion.

In this chapter, we investigate how we utilise the visual feature extracted on the focal-plane of SCAMP-5 for 6 DoF VO. The contributions of our work are:

- An efficient binary feature Visual Odometry, BIT-VO, is the first 6-DoF visual odometry which utilises the FPSP. Given no prior information about the scene and no intensity information, our proposed method can accurately track the pose at 300 FPS, even in difficult situations where the state-of-the-art monocular SLAM fails.
- A novel binary-edge-based descriptor, which is small and is only 44-bit long. Using noisy features computed on the focal-plane of the SCAMP-5 image sensor, our system can track keypoints using this binary descriptor.
- Extensive evaluation of our system against measurements from a motion capture system, including difficult scenarios such as violently shaking the device 4-5 times a second. We also compare against ORB-SLAM to highlight how approaches that utilise a conventional camera struggle with rapid motions.

3.2 Background

This section provides a background and literature review on analog computation, vision sensors, and SCAMP-5 FPSP.

3.2.1 Analog Computation

Modern computers operate using digital signals, representing the data as multiple distinct binary bits. In contrast, analog processing circuits use continuous signals such as electrical voltage. Compared to digital processing, analog processing is advantageous in terms of speed, power dissipation, and total area consumed by the circuitry; however, the functionality of the system is determined by how the components are connected, and hence, the programming of such circuit is achieved by reconfiguring the signal path [Dudek and Hicks, 2000]. This is significantly less flexible when compared to the digital circuit, where the functionality is determined by the *software* and requires no modification to the *hardware* to perform a wide range of tasks.

3.2.2 Vision Sensors

The mainstream imaging sensor technologies are Charge-Coupled Devices (CCDs) and Complementary Metal-Oxide-Semiconductor (CMOS). The main building blocks in both sensors are pixel array, readout system, and digital logic. The pixel array converts the captured photons to electric charges, and the readout system converts such analog signals to digital signals, moving the data off the focal-plane. Digital logic controls the system's operation, such as timing and driver. Of these three blocks, on most modern sensors, the readout system consumes over 50% of the total sensor power [Likamwa et al., 2016], while the pixel array accounts for a small fraction of the energy consumption [Kitamura et al., 2012]. These limitations motivated a design where analog processing is collocated with the pixel array, reducing and eliminating the work to be done by the ADC. Recent vision chips add a processing unit per pixel and operate as a Single Instruction Multiple Data (SIMD) computational device. Examples of such vision chips includes [Dominguez-Castro et al., 1997, Linan et al., 2002, Poikonen et al., 2009]; however, all chips have limited pixel count with less than 20,000 pixels, making them suitable only for applications where high pixel resolution is not required. SCAMP-5 (SIMD Current-mode Analog Matrix Processor), on the other hand, contains 256×256 pixels, which is closer to the resolution of a camera used in applications such as SLAM (MonoSLAM [Davison et al., 2007] used 320×240 resolution hand-held camera).

3.2.3 SCAMP-5 FPSP

A conventional camera is a 2D array of light-sensitive elements known as pixels. FPSP, also known as processor-per-pixel arrays (PPA) and cellular-processor arrays (CPA), adds a small processor per pixel on the same die [Zarándy, 2011]. Each pixel in SCAMP-5 [Carey et al., 2013b] combines a photodiode with a Processing Element (PE), where these PEs can execute an instruction simultaneously on their local data, resulting in SIMD parallel processing. Although the PEs operate on an analog signal, SCAMP uses "switched-current analog microprocessor" [Dudek and Hicks, 2000], which is an architecture similar to a general-purpose digital micro-processor (where there are instructions from the digital controller, registers, ALU, I/O); however, all operations are

analog. This allows the analog circuit to be software reprogrammable, increasing it's flexibility while being more efficient than the digital counterpart.

3.2.4 Visual Odometry Using SCAMP-5

While several works utilise FPSP for visual odometry, none support full 6-DoF pose estimation. In [Bose et al., 2017], the proposed method is a feature-based VO algorithm, estimating yaw, pitch, and roll rotations, as well as the translation along the z-axis. They first extract edge features and then align them with a keyframe. The alignment is done via shift, scale, and rotation operations performed on the sensor-processor chip. The 4-DoF algorithm can run up to 1000 FPS under sufficient lighting. The algorithm was later extended and deployed on a UAV [Greatwood et al., 2018, McConville et al., 2020], fusing the IMU measurements. Furthermore, [Debrunner et al., 2019] introduced a VO algorithm that operates at 400-500 FPS, also capable of estimating 4 DoF. Employing a direct approach that utilises image intensity, their method divides the focal-plane into tiles to estimate the optic flow for each tile. They then determine the 4-DoF motion by solving a least square problem.

3.3 System Overview

Our main contribution is a 6-DoF monocular visual odometry, which operates in real-time at 300 FPS. An overview of our system flow is summarised in Figure 3.2. The initialisation is omitted for simplicity. Feature extractions are performed on SCAMP-5, while feature tracking and VO operate on the host device, such as a consumer-grade laptop. The system operates only using the binary edge image and corner coordinates without transferring any pixel intensity information (as shown in Figure 3.1). Despite using limited information, we demonstrate the feasibility of creating a robust VO system against rapid motion.

3.4 Feature Detection and Matching

This section outlines how features are detected on the FPSP device and how these features are matched against previous ones on the host device.

3.4.1 Feature Detection

On the FPSP, we perform FAST keypoint and binary edge detection, which are computed at a high frame rate of 330 FPS on the focal-plane. An existing implementation of FAST Keypoint Detector for SCAMP-5 [Chen et al., 2017] is used and for edge detection, the magnitude of the image gradient is thresholded to find edges [Bose et al., 2017]. For each frame, up to 1000 corner features are detected and are read as pixel coordinates using an event-readout. The whole 256×256

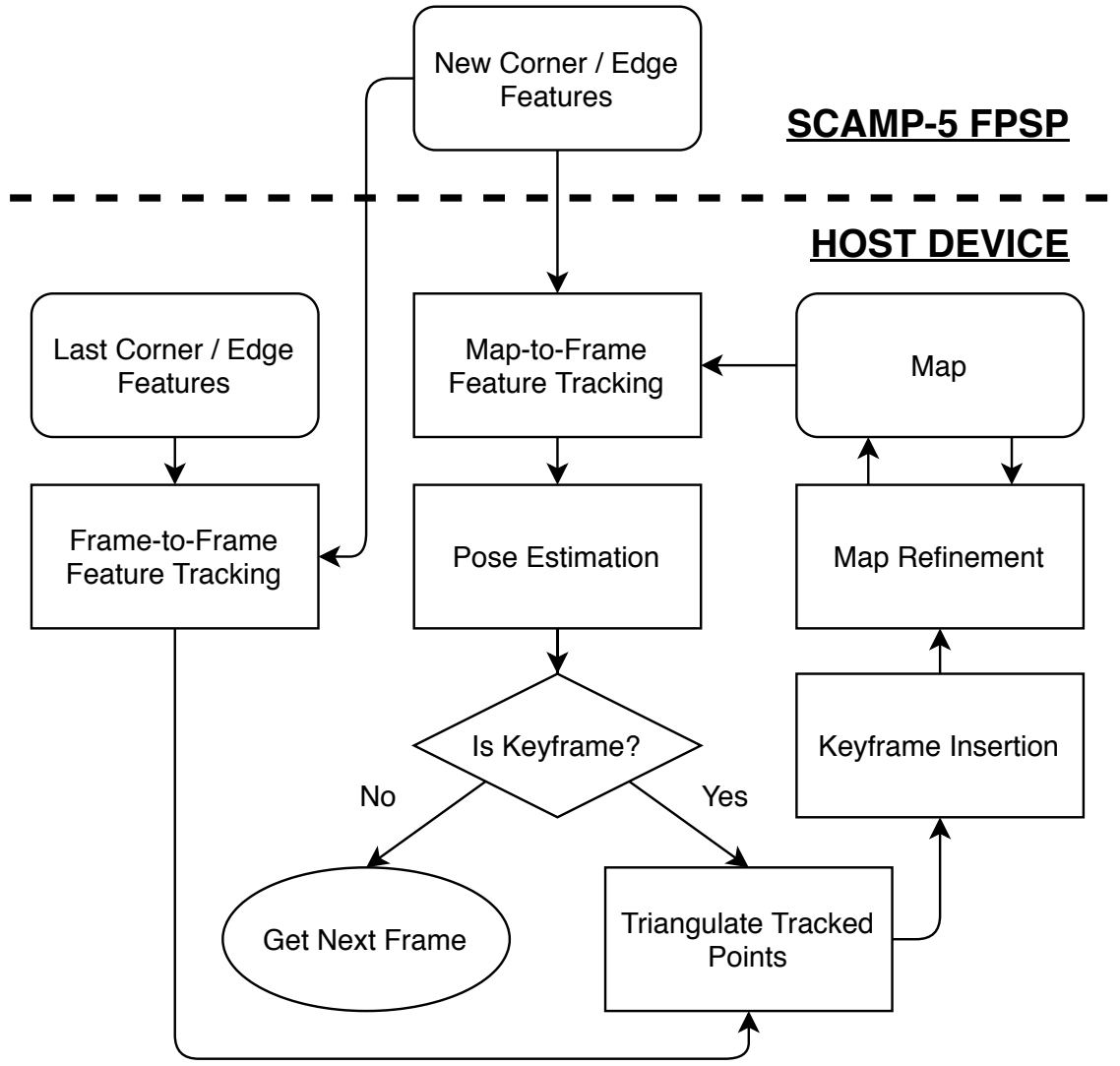


Figure 3.2: Tracking and Mapping pipeline. The pipeline runs on an FPSP and a host device, minimising data flow from the sensor to the host device,

bit binary image is transferred for the binary edge image rather than the pixel coordinates. In SCAMP-5, coordinates are expressed as an 8-bit pair. Hence, event read-outs are only efficient if the number of events $N_{events} < 4096$. This is only 6.25% of all the available pixels, and we found that, in most cases, the edge image exceeds this threshold.

3.4.2 Feature Matching

Matching the corner features extracted from SCAMP-5 across multiple frames is challenging for two reasons:

- Feature extraction suffers from noise in analog computation.
- Multiple features are detected per visual corner.

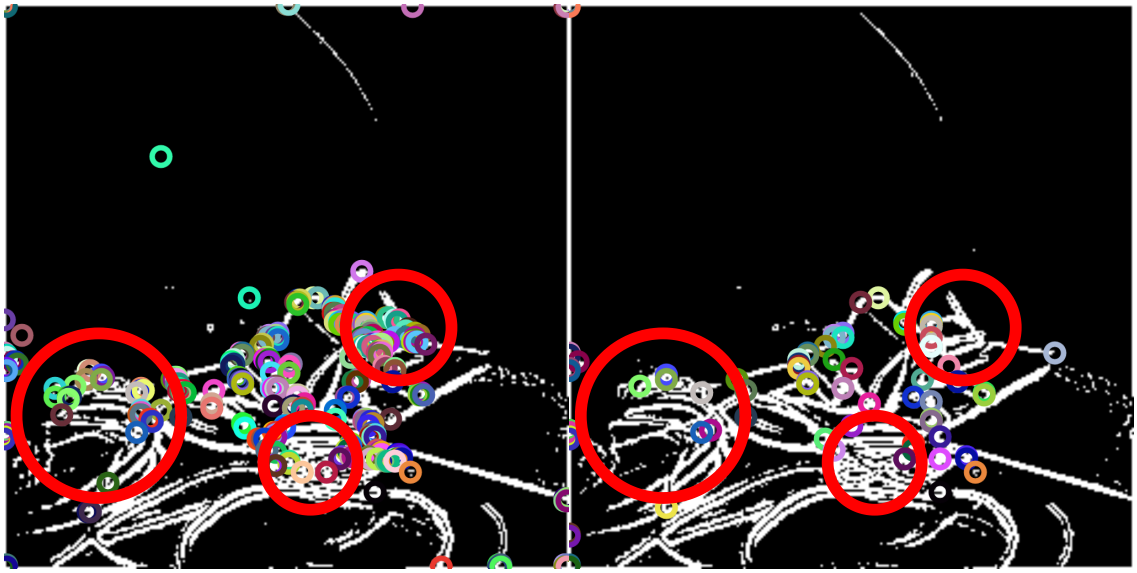


Figure 3.3: Illustration of the effect of noisy analog computation. Between two consecutive frames, many corners appear and disappear. The device was mounted on a tripod to ensure stability across multiple frames.

Due to the noisy nature of analog computation, corners are not repeatably extracted at every frame as shown in Figure 3.3, causing incorrect data association if a naive method such as the nearest neighbour matching is used. Since the features disappear between frames, we need multi-frame matching. Although [Chen et al., 2017] performs non-maximal suppression, the suppressed corners are inconsistent across multiple frames, so we require a method to reliably isolate features from each other for each visual corner. To address this problem, we introduce a binary descriptor that can be used for feature matching.

3.4.3 Local Binary Descriptors from Edges

We propose a feature descriptor that only uses the local binary edge information to establish robust correspondences across multiple feature frames. Our descriptor is tiny – only 44-bit long in length; thus, it is space-efficient and is fast to compute. Unlike other binary descriptors such as LBP [Ojala et al., 2002], BRIEF [Calonder et al., 2010], and BRISK [Leutenegger et al., 2011], we do not have access to the image intensity information. As shown in Figure 3.4, three independent rings $\{r1, r2, r3\}$ are formed around a corner of interest, and each element of the ring contains a bit from the corresponding pixel of the binary edge image. We use a 7×7 patch to store them with a single 64-bit unsigned integer, making it efficient to create and compare by bitwise manipulation. Each feature's orientation is computed to add a rotation invariance to our descriptor. Assuming a coordinate frame with the origin set to the corner feature of interest, the intensity gradient magnitude $G(x, y)$ [Rosin, 1999] is used to compute the orientation:

$$\theta = \tan^{-1} \frac{\sum_{x,y} yG(x, y)}{\sum_{x,y} xG(x, y)}, \tag{3.1}$$

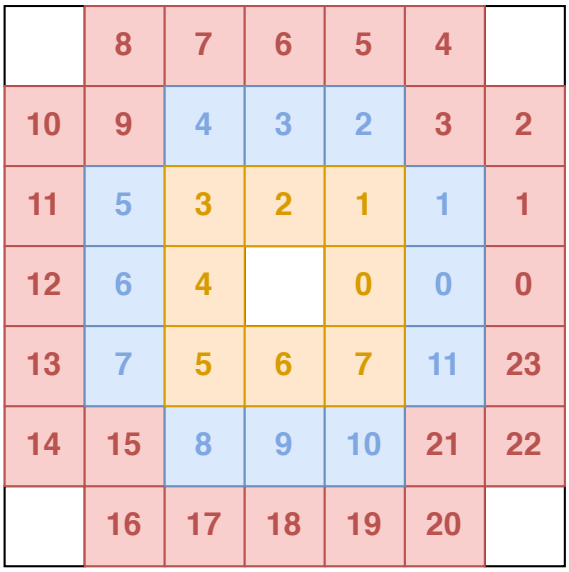


Figure 3.4: Descriptor sampling pattern. Different colours denote a different ring, and indices correspond to the bit index.

where x, y are the coordinates of the 7×7 patch. Since the gradient image is binarised, we approximate by setting $G(x, y)$ to 1 if image point (x, y) is classified as an edge, and setting it to 0 otherwise. Each ring's rotation invariance is achieved by applying a circular shift to the elements based on the orientation θ [Ojala et al., 2002]. At each ring, the number of bits to rotate is determined by:

$$R(\theta, r) = \left\lfloor \frac{\theta \cdot N_r}{2\pi} \right\rceil \tag{3.2}$$

where $\lfloor \cdot \rceil$ is the floor operation, $r \in \{r1, r2, r3\}$ and N_r is the number of elements in the given ring. The descriptor d is computed by:

$$d = \left(r1' << (N_{r2} + N_{r3})\right) \mid \left(r2' << N_{r3}\right) \mid r3' \tag{3.3}$$

where $<<$ operator is bit-wise shift, and $|$ operator is bit-wise or. r' denotes a ring compensated for the rotation using Equation 3.2. The descriptors are compared using the Hamming distance, performed efficiently using Streaming SIMD Extensions (SSE) instructions. Although our descriptors are not scale-invariant, we found them to be sufficient for small indoor scenes.

3.4.4 Frame-to-Frame Matching

Our system's high frame rate enables efficient frame-to-frame feature matching. Given frames, $\{F_1, \ldots, F_n\}$, a local neighbourhood around a feature in F_i is matched against features in F_{i+1}. Similarly, features in F_{i+1} are matched against features in F_{i+2}. By following these matches, features in F_i can be matched against any arbitrary frames as long as they remain in sight. We take advantage of the small inter-frame motion and only search a small radius of $3-5$ pixels to find correspondence. Here, we select the corner from the search radius that minimises the Hamming distance as a candidate match. If the descriptor distance to the candidate exceeds a threshold, the candidate does not form a correspondence, and in our implementation, the threshold is set to 10.

3.4.5 Map-to-Frame Matching

All visible map points are projected onto the image plane to find correspondences. Again, only a small radius is searched. Each map point stores multiple descriptors as they are observed across multiple keyframes. Following ORB-SLAM2 [Mur-Artal and Tardós, 2017], we select the most descriptive descriptor for each map point by finding the descriptor that minimises the median distance to all other descriptors.

3.5 Visual Odometry

This section summarises the implementation details of our VO system; however, it is kept brief as it is very similar to the standard VO systems like PTAM [Klein and Murray, 2007]. A set of 3D map points of the scene is used to estimate the pose of SCAMP-5 by minimising the reprojection error. After every keyframe insertion, we perform structure-only bundle adjustment [Strasdat et al., 2012] to refine the 3D map. These non-linear problems are solved using the Levenberg-Marquardt algorithm, which is implemented using Ceres Solver [Agarwal et al., 2010]. The non-linear optimisation converges with little iterations as the inter-frame motion is small.

3.5.1 Pose Estimation

Given a set of 3D map points and their correspondences on an image plane, poses can be estimated by minimising the reprojection error, which can be formulated as [Strasdat et al., 2012]:

$$\mathbf{T}_{CW} = \arg\min_{\mathbf{T}_{CW}} \frac{1}{2} \sum_i \rho \left(\| \mathbf{u}_i - \pi(\mathbf{T}_{CW} \cdot {}_W\mathbf{p}_i) \|^2 \right) , \tag{3.4}$$

where the error between the projected 3D points $\pi(\mathbf{T}_{CW} \cdot {}_W\mathbf{p}_i)$ and the corresponding feature coordinates $\mathbf{u}_i$ is minimised. $\rho(\cdot)$ is the Huber loss function, which reduces the effect of outlying data [Zhang, 1997]. Unlike PTAM [Klein and Murray, 2007] or ORB-SLAM [Mur-Artal and Tardós, 2017], the constant velocity model is not used in pose estimation.

3.5.2 Bootstrapping

The 5-point algorithm [Nistér, 2004] with RANSAC [Fischler and Bolles, 1981] is used to perform bootstrapping. This gives a relative pose estimate and is used to triangulate the initial 3D map. During the initialisation process, features in the reference frame are tracked using frame-to-frame tracking until sufficient disparities exist. Disparities are computed by taking the median of the feature's pixel displacements. If the disparity is greater than 20 pixels, relative pose estimation and triangulation are attempted. Upon triangulation, if any 3D map point has a parallax of fewer than 5 degrees or is behind either of the two cameras, it is removed from the map. We only successfully initialise the system once 100 map points are triangulated; otherwise, we will attempt this bootstrapping repeatedly.

3.5.3 Keyframe Selection

To determine which frames are suitable as keyframes, we adopt a selection process similar to PTAM [Klein and Murray, 2007] and SVO [Forster et al., 2014], [Forster et al., 2016]. This process is based on the camera displacement relative to the scene's depth. A frame is designated as a keyframe if it meets the following criteria:

- At least 200 frames have elapsed since the previous keyframe insertion.
- At least 50 features are tracked.
- Euclidean distances between the current frame and all the other keyframes are greater than 12% of the median scene depth.

When a frame is selected as a keyframe, first, 2D-3D correspondences are established through the projection of the map points into the image plane. This links the map point to the keyframes that were observed. If some features have not yet been triangulated, the Frame-to-Frame tracker is checked to see if any matches satisfy the epipolar constraint. If fewer than 30 matches are found, brute-force matching of all the features is performed between the current and the last keyframe. This ensures that a sufficient number of map points are generated with every keyframe insertion.

3.6 Experiments

We have evaluated our proposed system against ground-truth data captured using a Vicon motion-capture system. As our method is a monocular VO, the estimated trajectory is scaled and aligned to the ground truth data before computing the Absolute Trajectory Error (ATE) [Sturm et al., 2012]. All experiments are conducted using SCAMP-5 [Carey et al., 2013b]. SCAMP-5 does not record raw-intensity images because, in this case, it would act as a conventional camera and operate at a low frame rate. Thus, a direct comparison against other VO/SLAM using a monocular camera or SCAMP-5 is not possible. Instead, a webcam that operates at 20 FPS was attached to SCAMP-5 as a qualitative demonstration of how SLAM systems such as ORB-SLAM2 [Mur-Artal and Tardós, 2017] which uses conventional camera loses track under dynamic motions. Since the two cameras' fields of view are different, best efforts were made to ensure that both devices observed the same scene. All host computations were made on a laptop with a 4-core Intel i7-6700HQ CPU at 2.60GHz. Mapping and tracking used a single core, and visualisation and communication with SCAMP-5 used an extra core each.

We evaluate our system against four recordings: Long, Rapid Shake, Jumping, and Circle sequences. The test scene consisted of tabletop objects like desktop monitors and books. Videos of the live running system are available on the project page[2].

[2] https://rmurai.co.uk/projects/BIT-VO/

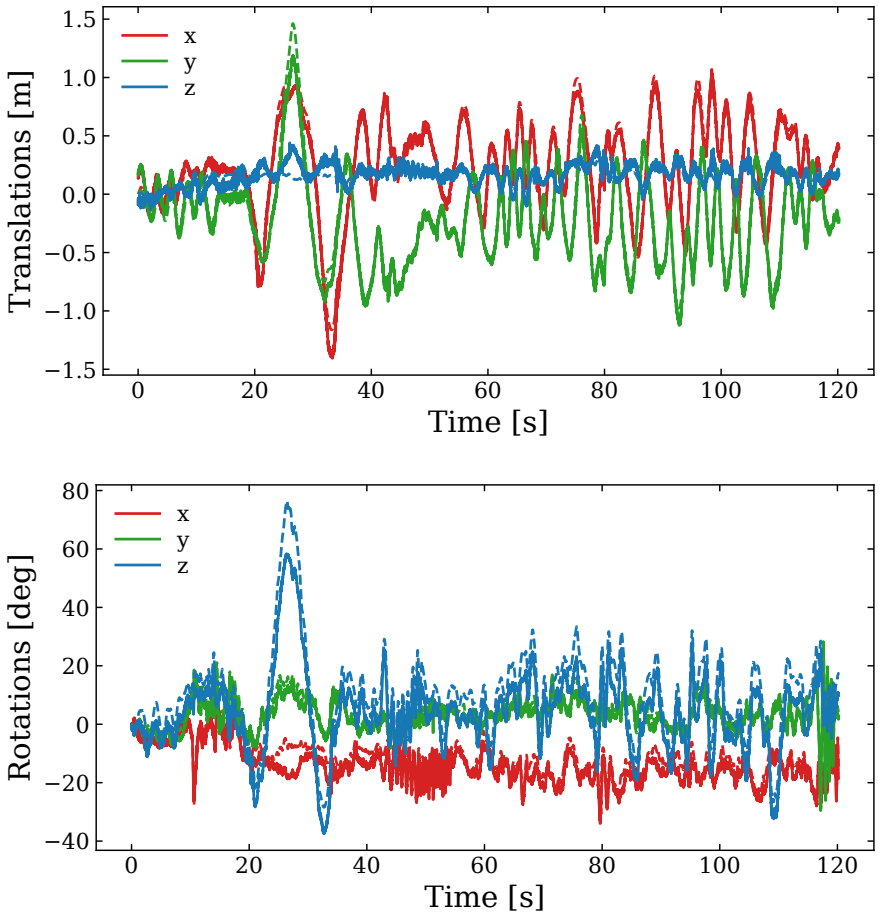


Figure 3.5: **Top:** Estimated x, y, z translations for "Long" sequence. **Bottom:** Estimated x, y, z rotations for "Long" sequence. Solid lines show our estimate and dotted line are the ground truth.

Table 3.1: Absolute Trajectory Error of different sequences, computed using evo [Grupp, 2017]. The total length of the trajectory, Root Mean Square Error and Median Error are reported.

Sequence	Length[m]	RMSE [m]	Median [m]
Long	68.5	0.108	0.078
Rapid Shake	5.6	0.015	0.011
Jumping	32.9	0.056	0.040
Circle	38.3	0.128	0.084

Table 3.2: Absolute Trajectory Error comparison of using our proposed descriptor and using rotated BRIEF, computed using evo [Grupp, 2017]. The total length of the trajectory, Root Mean Square Error and Median Error are reported.

Descriptor	Length[m]	RMSE [m]	Median [m]
Ours	38.3	0.128	0.084
Rotated BRIEF	38.3	0.123	0.107

3.6.1 Accuracy and Robustness

The "Long" test sequence involves repeatedly traversing a test area for a total of 68.5m, with many features appearing and disappearing from the view of SCAMP-5. The translation and rotation of

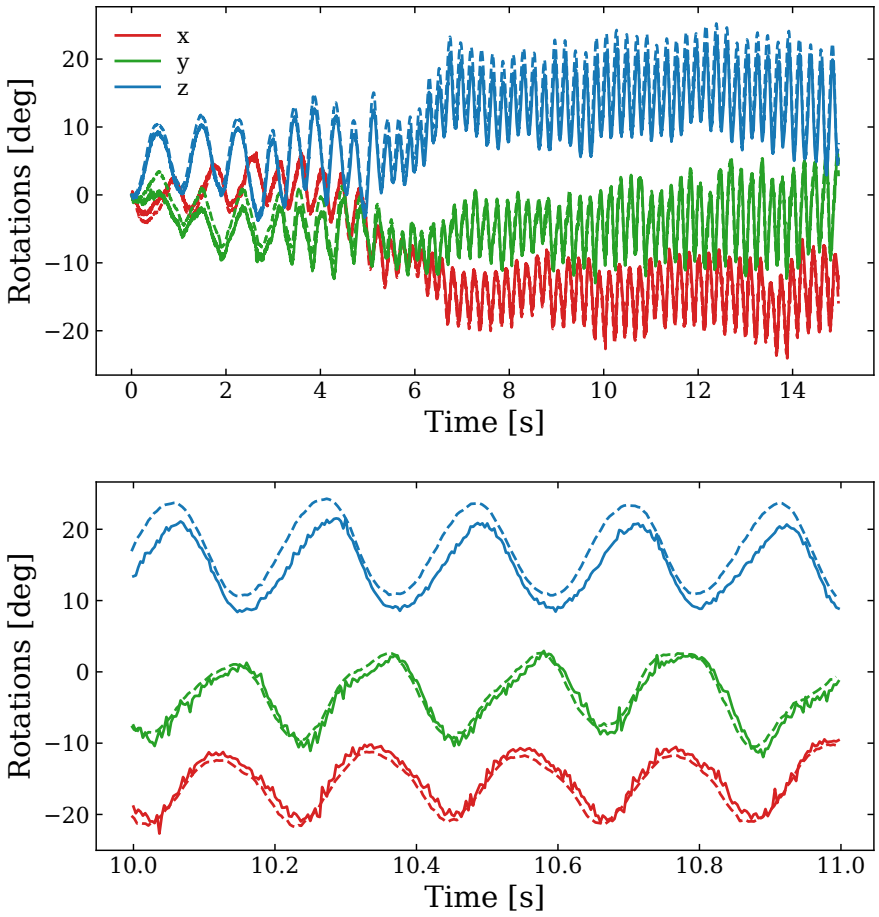


Figure 3.6: **Top:** Estimated x, y, z rotations for "Rapid Shake" sequence. **Bottom:** Close-up view of rotation estimates for "Rapid Shake" sequence. Solid lines show our estimate and dotted line are the ground truth. Our method is able to track rapid rotations accurately.

our system over time are depicted in Figure 3.5. We report the ATE error of all of our runs in Table 3.1.

Similar to a 4-DoF VO for SCAMP-5 [Bose et al., 2017], our system is able to track violent rotations, as shown in Figure 3.6. The system was subject to 4-5 shakes per second but was able to track rotations along all three axes successfully and accurately.

3.6.2 Comparison Against Visual SLAM

This section presents the advantage of our high frame rate VO, running at 300 FPS, compared with ORB-SLAM2 [Mur-Artal and Tardós, 2017] running on images from a conventional camera. For a fair comparison, in all runs, the images were resized and cropped to 256×256 pixels to match the resolution of SCAMP-5. An inherent limitation of SCAMP-5 is that it is not possible to record image intensity, as only the processed features are transferred to the host device. Any attempt to save images will require ADC and slow down the entire pipeline; thus, for comparisons against other SLAM systems, the intensity images must be captured using an external conventional

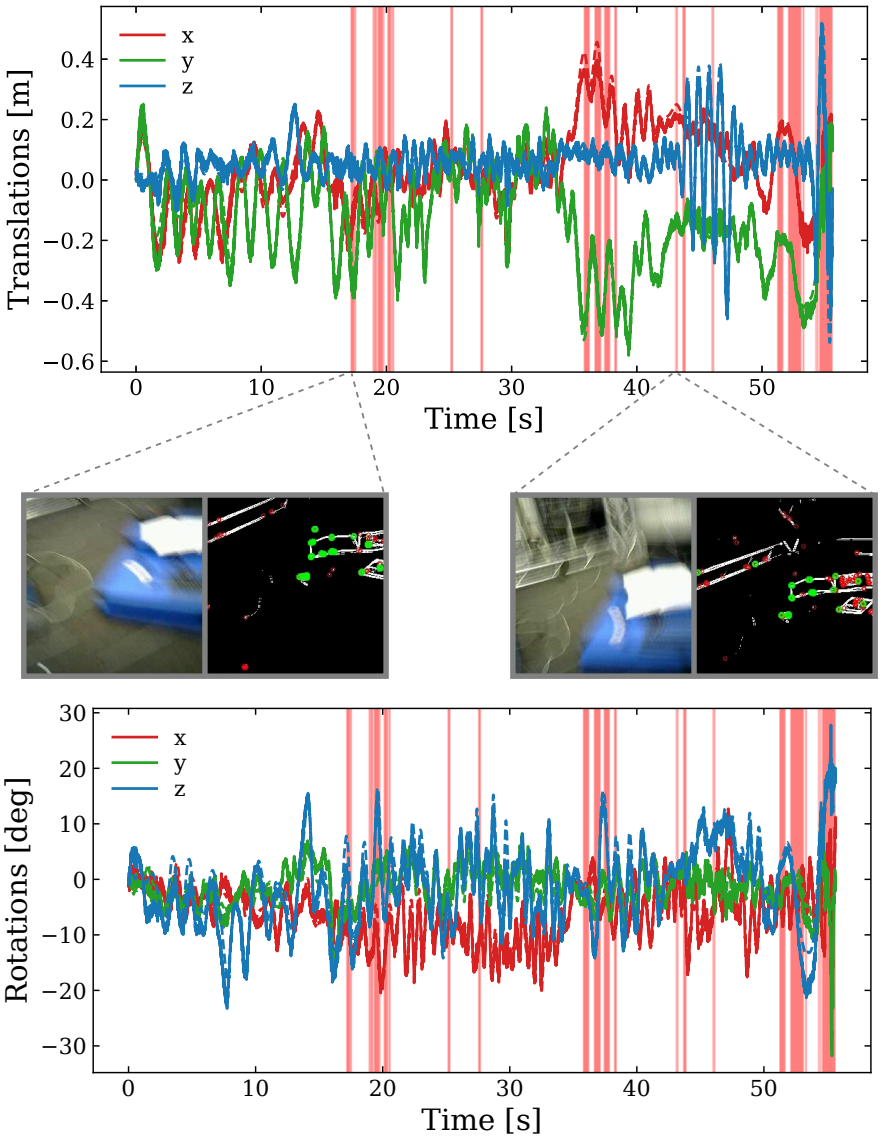


Figure 3.7: **Top:** Estimated x, y, z rotations for "Jumping" sequence. **Bottom:** Estimated x, y, z rotations for "Jumping" sequence. Solid lines show our estimate and dotted line are the ground truth. The pink region indicates that the ORB-SLAM2 lost track due to rapid motion.

camera.

In the "Jumping" sequence, the device is subject to violent translational motions, including sudden ∼80cm change in the z-axis caused by jumping as shown in seconds 42-48 of Figure 3.7. This comparison highlights the clear advantage of operating at 300 FPS. The pink highlighted regions in Figure 3.7 are where ORB-SLAM lost track (in VO mode, it recovers using relocalisation). As shown, the images captured by the conventional camera suffer from severe motion blur, while the features from SCAMP-5 do not.

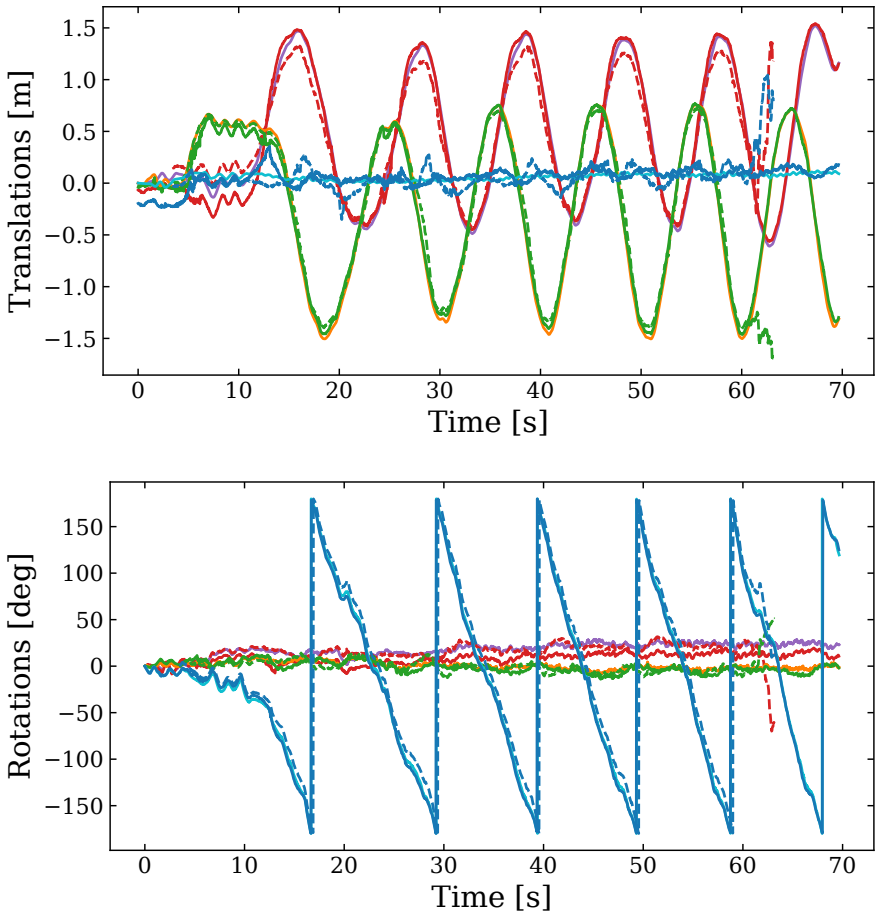


Figure 3.8: **Top:** Estimated x, y, z translation for "Circle" sequence. **Bottom:** Estimated x, y, z rotations for "Circle" sequence. Solid lines show results from using our proposed descriptor, while dotted lines used rotated BRIEF. The estimated data x, y, z is plotted using red, green, blue and the ground truth data x, y, z is plotted using purple, orange, cyan respectively. Note rotations along z-axis wraps as full 360 degrees loops are made.

3.6.3 Comparison Against Other Descriptors

Another possible choice of descriptor would have been to employ other binary descriptors like BRIEF [Calonder et al., 2010] or BRISK [Leutenegger et al., 2011]. However, these methods construct the descriptor by comparing pixel intensities. To compare against our approach, the BRIEF descriptor was modified by using XOR operation instead of pixel intensity comparison. To achieve rotation invariance, we adopt the same methodology as ORB [Rublee et al., 2011], where the feature's orientation is calculated using Equation 3.1.

To compare our descriptor against BRIEF, we have recorded the output features from SCAMP-5. The Vicon room was explored in a circular motion while the camera was pointing toward the centre of the room. A modified version of 256-bit long rotated BRIEF from OpenCV [Bradski, 2000] was used for the experiments. Figure 3.8 shows no major differences in the two approaches, apart from 60 seconds onward where VO using rotated BRIEF fails. Figure 3.9 depicts the 3D trajectories of our approaches together with the ground truth. We notice that there are high-frequency

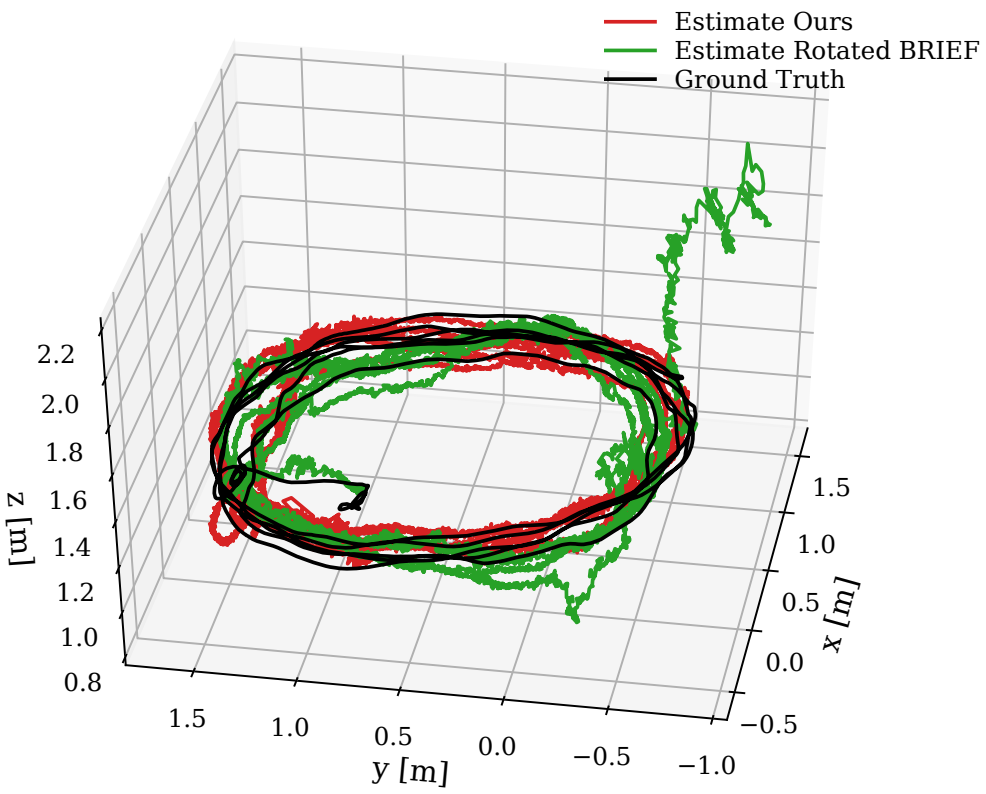


Figure 3.9: Estimated 3D trajectory of "Circle" sequence using our proposed method: our pipeline (in red), rotated BRIEF descriptors (in green), and ground truth (in black).

noises present in our trajectories. The SCAMP-5 camera's low resolution means there is a large round-off error in the pixel positions of the features. Furthermore, due to the noise present in the analog computation of each frame, a different set of corners is extracted for the same visual scene, leading to incorrect feature correspondences and, thus, a shaky trajectory. Table 3.2 compares the absolute trajectory error using two descriptors. To ensure a fair comparison, measurements after 58 seconds were excluded for rotated BRIEF when it failed to track the trajectory. The results show no significant difference in the tracking accuracy when using either descriptor. However, our descriptor has a significant advantage in terms of computational efficiency, as shown in Figure 3.10. The runtime for computing the descriptors per frame was measured offline over ten iterations for the "Circle" sequence for both our descriptor and rotated BRIEF. The median runtime for our approach was more than five times faster than rotated BRIEF.

3.6.4 Runtime Evaluation

A breakdown of the runtime of the motion estimation that occurs on the host device is provided in Figure 3.10. The timing is measured offline over ten iterations of the "Circle" sequence. Our motion estimation is highly efficient, and the median time required to estimate the pose is 1.10 ms when executed offline, which translates to a frame rate of over 900 FPS. Our system does not

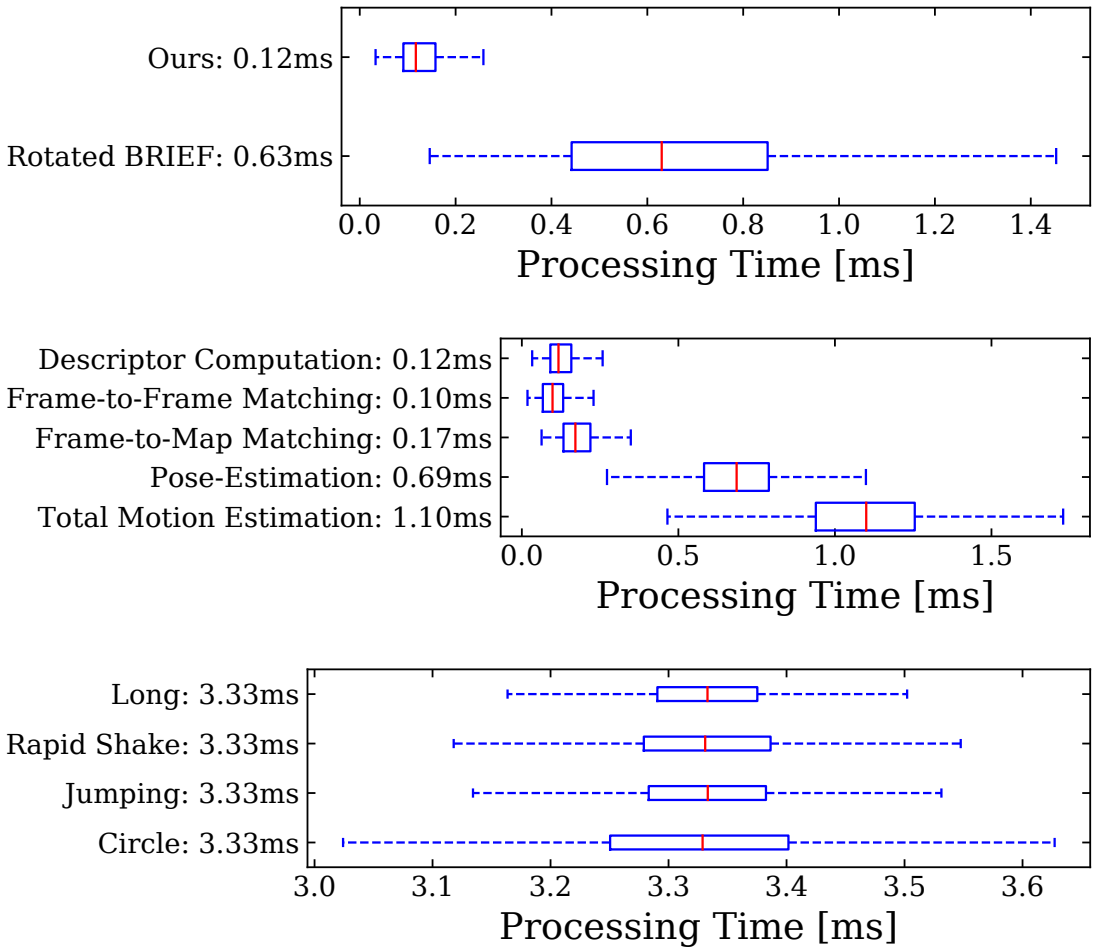


Figure 3.10: Runtime breakdown of the system. **Top:** Comparison of the processing time of our descriptor against rotated BRIEF. **Middle:** Breakdown of the processing time required by our motion estimation. **Bottom:** Processing time per frame while running the system online on different sequences. Note that the bottleneck is SCAMP-5, which outputs features at 300 FPS.

separate map refinement onto different threads during keyframe insertion. The median processing time for keyframe insertion is 3.17ms, with 2.22ms and 3.98ms at 0.25 and 0.75 quantiles, respectively. The keyframe insertion combined with motion estimation exceeds the time budget of 3.33ms when operating at 300 FPS. However, the excess is resolved within one or two frames. For latency-critical applications, offloading the keyframe insertion onto a different thread is possible. The runtime of the different sequences when operating the system live is also reported. For stable frame rates, we execute SCAMP-5 at 300 FPS, not at full capacity of 330 FPS. Execution of the feature extraction on SCAMP-5 is our bottleneck, which limits the overall frame rate of BIT-VO. The rest of the pipeline can run at a much higher frame rate; thus, our approach is applicable to the next-generation FPSP devices, which may have much faster computation.

Finally, the “Circle” sequence has the largest inter-quantile range, as it required more keyframe insertions compared to other sequences.

3.7 Conclusion

We presented BIT-VO, a VO framework capable of operating at 300 FPS using binary edges and corners computed on the focal-plane. Our system is simple and minimal yet sufficient to work in challenging conditions, highlighting the advantage of operating at high effective frame rates. The proposed pipeline implemented a robust feature-matching scheme using small 44-bit long descriptors. FPSP's analog computation introduces noise to the values, but the proposed method can distinguish the noisy features. We demonstrated that processing data in the focal-plane and limiting data movement only to important features could increase the frame rate of VO system and achieve low latency without consuming much energy.

While SCAMP-5 offers many promising potentials, the fill factor of the sensor is low because the processing elements are collocated with the photodiode on each pixel. This means that BIT-VO requires the scene to be well-illuminated, and all of our experiments were conducted in well-lit rooms. However, this limitation could potentially be addressed by using modern fabrication technologies, such as stacked CMOS. Additionally, the instruction set and registers available on the device are limited. Ideally, rather than performing feature matching on the host device, one would perform more advanced computation (e.g., feature matching or dense optical flow) directly on the focal-plane. However, these are challenging to implement on the current iteration of the SCAMP-5 camera. We hope this work will inform the design of future FPSP devices with higher computational capability, light sensitivity, and pixel count. The FPSP device's programmable nature, in contrast to, for example, event cameras, offers the prospect of higher accuracy and enhanced robustness through greater adaptivity.

CHAPTER 4

Gaussian Splatting SLAM

Contents

Work within this chapter results from a close collaboration with Hidenobu Matsuki, leading to a paper: **Gaussian Splatting SLAM**, Hidenobu Matsuki*, Riku Murai*, Paul H.J. Kelly, Andrew J. Davison, CVPR 2024 (*denotes equal contribution) [Matsuki et al., 2024].

This chapter presents the first application of 3D Gaussian Splatting in monocular SLAM, the most fundamental but the hardest setup for Visual SLAM. Our method, which runs live at 3 FPS, utilises Gaussians as the only 3D representation, unifying the required representation for accurate,

Figure 4.1: From a single monocular camera, we reconstruct a high-fidelity 3D scene live at 3 FPS. For every incoming RGB frame, 3D Gaussians are incrementally formed and optimised together with the camera poses. We show both the rasterised Gaussians (left) and Gaussians shaded to highlight the geometry (right). Notice the details and the complex material properties (*e.g.* transparency) captured. Thin structures such as wires are accurately represented by numerous small, elongated Gaussians, and transparent objects are effectively represented by placing the Gaussians along the rim. Our system significantly advances the fidelity a live monocular SLAM system can capture.

efficient tracking, mapping, and high-quality rendering. Designed for challenging monocular settings, our approach is seamlessly extendable to RGB-D SLAM when an external depth sensor is available. Several innovations are required to continuously reconstruct 3D scenes with high fidelity from a live camera. First, to move beyond the original 3DGS algorithm, which requires accurate poses from an offline Structure from Motion (SfM) system, we formulate camera tracking for 3DGS using direct optimisation against the 3D Gaussians and show that this enables fast and robust tracking. Second, by utilising the explicit nature of the Gaussians, we introduce geometric verification and regularisation to handle the ambiguities occurring in incremental 3D dense reconstruction. Finally, we introduce a full SLAM system which not only achieves state-of-the-art results in novel view synthesis and trajectory estimation but also reconstruction of tiny and even transparent objects.

4.1 Introduction

A long-term goal of online reconstruction with a single moving camera is near-photorealistic fidelity, which will surely allow new levels of performance in many areas of Spatial AI and robotics as well as opening up a whole range of new applications. While we increasingly see the benefit of applying powerful pre-trained priors to 3D reconstruction, a key avenue for progress is still the invention and development of core 3D representations with advantageous properties. Many "layered" SLAM methods exist which tackle the SLAM problem by integrating multiple different 3D representations or existing SLAM components; however, the most interesting advances are when a new unified dense representation can be used for all aspects of a system's operation: local representation of detail, large-scale geometric mapping and also camera tracking by direct alignment.

We present the first online visual SLAM system based solely on the 3D Gaussian Splatting (3DGS)

representation [Kerbl et al., 2023] recently making a big impact in offline scene reconstruction. In 3DGS a scene is represented by a large number of Gaussian blobs with orientation, elongation, colour and opacity. Other previous world/map-centric scene representations used for visual SLAM include occupancy or Signed Distance Function (SDF) voxel grids [Newcombe et al., 2011a]; meshes [Schöps et al., 2019]; point or surfel clouds [Keller et al., 2013, Schöps et al., 2019]; and recently neural fields [Sucar et al., 2021]. Each of these has disadvantages: grids use significant memory and have bounded resolution, and even if octrees or hashing allow more efficiency they cannot be flexibly warped for large corrections [Vespa et al., 2018, Nießner et al., 2013]; meshes require difficult, irregular topology to fuse new information; surfel clouds are discontinuous and difficult to fuse and optimise; and neural fields require expensive per-pixel raycasting to render. We show that 3DGS has none of these weaknesses. As a SLAM representation, it is most similar to point and surfel clouds and inherits their efficiency, locality and ability to be easily warped or modified. However, it also represents geometry in a smooth, continuously differentiable way: a dense cloud of Gaussians merge together and jointly define a continuous volumetric function. Crucially, modern graphics card's increased compute capabilities and large VRAM (video random access memory) allows for fast radix-based sorting and software rasterisation which means that a large number of Gaussians can be efficiently rendered via "splatting" rasterisation, up to 200 FPS at 1080p. This rapid, differentiable rendering is integral to the tracking and map optimisation loops in our system.

Up until now, the 3DGS representation has only been used in offline systems for 3D reconstruction with known camera poses, and we present several innovations to enable online SLAM. We first derive the analytic Jacobian on Lie group of camera pose with respect to a 3D Gaussian map. We show that this can be seamlessly integrated into the existing differentiable rasterisation pipeline to optimise camera poses alongside scene geometry. Second, we introduce a novel Gaussian isotropic shape regularisation to ensure geometric consistency, which we have found is important for incremental reconstruction. Third, we propose a novel Gaussian resource allocation and pruning method to keep the geometry clean and enable accurate camera tracking. Our experimental results demonstrate photorealistic online local scene reconstruction, as well as state-of-the-art camera trajectory estimation and mapping for larger scenes compared to other rendering-based SLAM methods. Our method works with only monocular input, which is one of the most challenging scenarios in SLAM. To highlight the intrinsic capability of 3D Gaussian for camera localisation, our method does not use any pre-trained monocular depth predictor or other existing tracking modules but relies solely on RGB image inputs in line with the original 3DGS. Since this is one of the most challenging SLAM scenarios, we also show our method can easily be extended to RGB-D SLAM when depth measurements are available.

In summary, our contributions are as follows:

- The first near real-time SLAM system which works with a 3DGS as the only underlying scene representation, which can handle monocular only inputs.
- Novel techniques within the SLAM framework, including the analytic Jacobian on Lie

group for direct camera pose estimation, isotropic regularisation of the Gaussian shape, and geometric verification.

- Extensive evaluations on a variety of datasets both for monocular and RGB-D settings, demonstrating competitive performance, particularly in real-world scenarios.

4.2 Related Work

Dense visual SLAM focuses on reconstructing detailed 3D maps, unlike sparse SLAM methods which excel in pose estimation [Mur-Artal et al., 2015, Engel et al., 2017, Forster et al., 2014] but typically yield maps useful mainly for localisation. In contrast, dense SLAM creates interactive maps beneficial for broader applications, including AR and robotics. Dense SLAM methods are generally divided into two primary categories: Frame-centric and Map-centric.

4.2.1 Frame-centric SLAM

In frame-centric SLAM, photometric errors are minimised across consecutive frames by jointly estimating per-frame depth and frame-to-frame camera motion. Frame-centric approaches [Teed and Deng, 2021, Czarnowski et al., 2020] are efficient, as individual frames host local rather than global geometry (*e.g.* depth maps), and are attractive for long-session SLAM, but if a dense global map is needed, it must be constructed on demand by assembling all of these parts which are not necessarily fully consistent.

4.2.2 Map-centric SLAM

Map-centric SLAM uses a unified 3D representation in contrast across the SLAM pipeline, enabling a compact and streamlined system. Compared to purely local frame-to-frame tracking, a map-centric approach leverages global information by tracking against the reconstructed 3D consistent map. We've reviewed the use of classical map-centric 3D representations such as voxels and surfels for SLAM in Section 1.7. Recently, in addition to classical graphic primitives, neural network-based map representations are a promising alternative. iMAP [Sucar et al., 2021] demonstrated the interesting properties of neural representation, such as sensible hole filling of unobserved geometry. Many recent approaches combine the classical and neural representations to capture finer details [Zhu et al., 2022, Sandström et al., 2023, Johari et al., 2023, Zhu et al., 2024, Rosinol et al., 2023]; however, a large amount of computation required for neural rendering makes the live operation of such systems challenging.

Our method adopts a Map-centric approach, utilising 3D Gaussians as the only SLAM representation. Similar to surfel-based SLAM, we dynamically allocate the 3D Gaussians, enabling us to model an arbitrary spatial distribution of the scene. Unlike other methods such as ElasticFusion [Whelan et al., 2015b] and PointFusion [Keller et al., 2013], however, by using differentiable

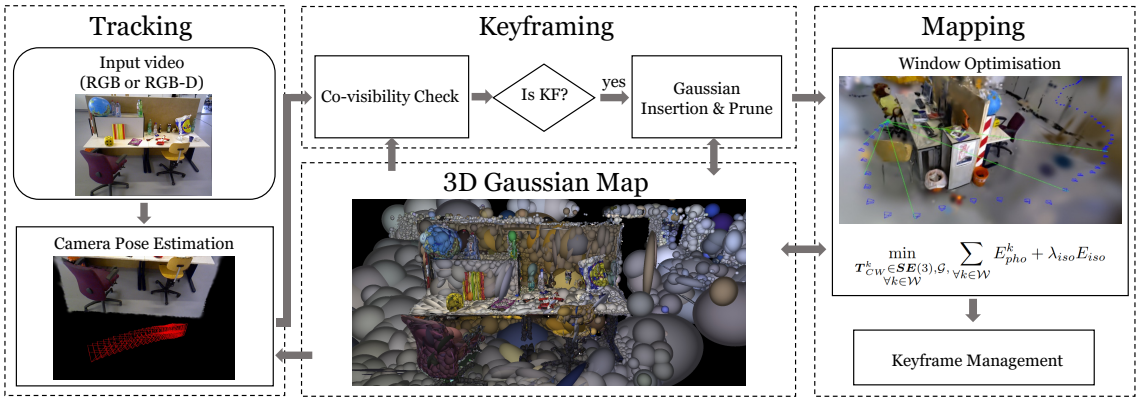


Figure 4.2: **SLAM System Overview:** Our SLAM system uses 3D Gaussians as the only representation, unifying all components of SLAM, including tracking, mapping, keyframe management, and novel view synthesis.

rasterisation, our SLAM system can capture high-fidelity scene details and represent challenging object properties by direct optimisation against information from every pixel.

4.3 Gaussian Splatting SLAM

Our main contribution is the full SLAM system, which uses 3DGS as the only SLAM representation. Here, we give an overview of 3DGS, how we perform camera pose optimisation against 3DGS, and the implementation details of the SLAM system. The overview of the system is summarised in Figure 4.2.

4.3.1 Gaussian Splatting

Our SLAM representation is 3DGS, mapping the scene with a set of anisotropic Gaussians $\mathcal{G}$. Each Gaussian $\mathcal{G}^i$ contains optical properties: colour c^i and opacity α^i. For continuous 3D representation, the mean $\boldsymbol{\mu}_W^i$ and covariance $\boldsymbol{\Sigma}_W^i$, defined in the world coordinate, represent the Gaussian's position and its ellipsoidal shape. For simplicity, we omit the spherical harmonics (SHs) representing view-dependent radiance. Since 3DGS uses volume rendering, explicit extraction of the surface is not required. Instead, by splatting and blending N Gaussians, a pixel colour $\mathcal{C}_p$ is synthesised:

$$\mathcal{C}_p = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \, . \tag{4.1}$$

3DGS performs rasterisation, iterating over the Gaussians rather than marching along the camera rays, and hence, free spaces are ignored during rendering. During rasterisation, the contributions of α are decayed via a Gaussian function, based on the 2D Gaussian formed by splatting a 3D Gaussian. The 3D Gaussians $\mathcal{N}(\boldsymbol{\mu}_W, \boldsymbol{\Sigma}_W)$ in world coordinates are related to the 2D Gaussians $\mathcal{N}(\boldsymbol{\mu}_I, \boldsymbol{\Sigma}_I)$ on the image plane through a projective transformation:

$$\boldsymbol{\mu}_I = \pi(\mathbf{T}_{CW} \cdot \boldsymbol{\mu}_W) \, , \boldsymbol{\Sigma}_I = \mathbf{J}\mathbf{W}\boldsymbol{\Sigma}_W\mathbf{W}^T\mathbf{J}^T \, , \tag{4.2}$$

where π is the projection operation and $\mathbf{T}_{CW} \in \mathbf{SE}(3)$ is the camera pose of the viewpoint. $\mathbf{J}$ is the Jacobian of the linear approximation of the projective transformation, and $\mathbf{W}$ is the rotational component of $\mathbf{T}_{CW}$. This formulation enables the 3D Gaussians to be differentiable, and the blending operation provides gradient flow to the Gaussians. Using first-order gradient descent [Kingma and Ba, 2015], Gaussians gradually refine both their optic and geometric parameters to represent the captured scene with high fidelity.

4.3.2 Camera Pose Optimisation

To achieve accurate tracking, we typically require at least 50 iterations of gradient descent per frame. This requirement emphasises the necessity of a representation with computationally efficient view synthesis and gradient computation, making the choice of 3D representation a crucial part of designing a SLAM system.

In order to avoid the overhead of automatic differentiation, 3DGS implements rasterisation with CUDA with derivatives for all parameters calculated explicitly. Since rasterisation is performance critical, we similarly derive the camera Jacobians explicitly.

To the best of our knowledge, we provide the first analytical Jacobian of $\mathbf{SE}(3)$ camera pose with respect to the 3D Gaussians used in EWA splatting [Zwicker et al., 2002] and 3DGS. This opens up new applications of 3DGS beyond SLAM.

We use Lie algebra to derive the minimal Jacobians, ensuring that the dimensionality of the Jacobians matches the degrees of freedom, eliminating any redundant computations. The terms of Equation 4.2 are differentiable with respect to the camera pose $\mathbf{T}_{CW}$; using the chain rule:

$$\frac{\partial \boldsymbol{\mu}_I}{\partial \mathbf{T}_{CW}} = \frac{\partial \boldsymbol{\mu}_I}{\partial \boldsymbol{\mu}_C} \frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\mathbf{T}_{CW}} , \tag{4.3}$$

$$\frac{\partial \boldsymbol{\Sigma}_I}{\partial \mathbf{T}_{CW}} = \frac{\partial \boldsymbol{\Sigma}_I}{\partial \mathbf{J}} \frac{\partial \mathbf{J}}{\partial \boldsymbol{\mu}_C} \frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\mathbf{T}_{CW}} + \frac{\partial \boldsymbol{\Sigma}_I}{\partial \mathbf{W}} \frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{T}_{CW}} . \tag{4.4}$$

where $\mathbf{T}_{CW}$ represents the 3D position of Gaussian in the camera coordinate. We take the derivatives on the manifold to derive minimal parameterisation. Borrowing the notation from [Solà et al., 2018], let $\mathbf{T} \in \mathbf{SE}(3)$ and $\tau \in \mathfrak{se}(3)$. We define the partial derivative on the manifold as:

$$\frac{\mathcal{D}f(\mathbf{T})}{\mathcal{D}\mathbf{T}} \triangleq \lim_{\tau \to 0} \frac{\mathrm{Log}(f(\mathrm{Exp}(\tau) \circ \mathbf{T}) \circ f(\mathbf{T})^{-1})}{\tau} , \tag{4.5}$$

With this, we derive the following:

$$\frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\mathbf{T}_{CW}} = \begin{bmatrix} \boldsymbol{I} & -[\boldsymbol{\mu}_C]_\times \end{bmatrix} , \frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{T}_{CW}} = \begin{bmatrix} \mathbf{0} & -[\mathbf{W}_{:,1}]_\times \\ \mathbf{0} & -[\mathbf{W}_{:,2}]_\times \\ \mathbf{0} & -[\mathbf{W}_{:,3}]_\times \end{bmatrix} , \tag{4.6}$$

where $[\cdot]_\times$ denotes the skew symmetric matrix of a 3D vector, and $\mathbf{W}_{:,i}$ refers to the ith column of the matrix. We provide the full derivation in Section 9.1.

4.3.3 Tracking

In tracking, only the current camera pose is optimised without updates to the map representation. In the monocular case, we minimise the following photometric residual:

$$E_{pho} = \left\| I(\mathcal{G}, \mathbf{T}_{CW}) - \bar{I} \right\|_1 , \tag{4.7}$$

where $I(\mathcal{G}, \mathbf{T}_{CW})$ renders the Gaussians $\mathcal{G}$ from $\mathbf{T}_{CW}$, and $\bar{I}$ is an observed image.

We further optimise affine brightness parameters for varying exposure and penalise non-edge or low-opacity pixels. When depth observations are available, we define the geometric residual as:

$$E_{geo} = \left\| D(\mathcal{G}, \mathbf{T}_{CW}) - \bar{D} \right\|_1 , \tag{4.8}$$

where $D(\mathcal{G}, \mathbf{T}_{CW})$ is depth rasterisation and $\bar{D}$ is the observed depth. Rather than simply using the depth measurements to initialise the Gaussians, we minimise both photometric and geometric residuals: $\lambda_{pho} E_{pho} + (1 - \lambda_{pho}) E_{geo}$, where λ_{pho} is a hyperparameter.

As in Equation 4.1, per-pixel depth is rasterised by alpha-blending:

$$\mathcal{D}_p = \sum_{i \in N} z_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) , \tag{4.9}$$

where z_i is the distance to the mean $\boldsymbol{\mu}_W$ of Gaussian i along the camera ray. We derive analytical Jacobians for the camera pose optimisation in a similar manner to Equation 4.3, Equation 4.4.

4.3.4 Keyframing

Since using all the images from a video stream to jointly optimise the Gaussians and camera poses online is infeasible, we maintain a small window $\mathcal{W}_k$ consisting of carefully selected keyframes based on inter-frame covisibility. Ideal keyframe management will select non-redundant keyframes observing the same area, spanning a wide baseline to provide better multiview constraints.

Selection and Management Every tracked frame is checked for keyframe registration based on our simple yet effective criteria. We measure the covisibility by measuring the intersection over the union of the observed Gaussians between the current frame i and the last keyframe j. If the covisibility drops below a threshold, or if the relative translation t_{ij} is large with respect to the median depth, frame i is registered as a keyframe. For efficiency, we maintain only a small number of keyframes in the current window $\mathcal{W}_k$ following the keyframe management heuristics of DSO [Engel et al., 2017]. The main difference is that a keyframe is removed from the current window if the overlap coefficient with the latest keyframe drops below a threshold.

Gaussian Covisibility An accurate estimate of covisibility simplifies keyframe selection and management. 3DGS respects visibility ordering since the 3D Gaussians are sorted along the camera ray. This property is desirable for covisibility estimation as occlusions are handled by design.

A Gaussian is marked to be visible from a view if used in the rasterisation and if the ray's accumulated α has not yet reached 0.5. This enables our estimated covisibility to handle occlusions without requiring additional heuristics.

Gaussian Insertion and Pruning At every keyframe, new Gaussians are inserted into the scene to capture newly visible scene elements and to refine the fine details. When depth measurements are available, Gaussian means $\boldsymbol{\mu}_W$ are initialised by back-projecting the depth. In the monocular case, we render the depth at the current frame. For pixels with depth estimates, $\boldsymbol{\mu}_W$ are initialised around those depths with low variance; for pixels without the depth estimates, we initialise $\boldsymbol{\mu}_W$ around the median depth of the rendered image with high variance.

In the monocular case, the positions of many newly inserted Gaussians are incorrect. While the majority will quickly vanish during optimisation as they violate multiview consistency, we further prune the excess Gaussians by checking the visibility amongst the current window $\mathcal{W}_k$. If the Gaussians inserted within the last 3 keyframes are unobserved by at least 3 other frames, we prune them out as they are geometrically unstable.

4.3.5 Mapping

The purpose of mapping is to maintain a coherent 3D structure and to optimise the newly inserted Gaussians. During mapping, the keyframes in $\mathcal{W}_k$ are used to reconstruct currently visible regions. Additionally, two random past keyframes $\mathcal{W}_r$ are selected per iteration to avoid forgetting the global map. Rasterisation of 3DGS imposes no constraint on the Gaussians along the viewing ray direction, even with a depth observation. This is not a problem when sufficient carefully selected viewpoints are provided (*e.g.* in the novel view synthesis case); however, in SLAM, where viewpoints are incrementally added, this causes many artefacts, making tracking challenging. We therefore introduce an isotropic regularisation:

$$E_{iso} = \sum_{i=1}^{|\mathcal{G}|} \left\| \mathbf{s}_i - \tilde{\mathbf{s}}_i \cdot \mathbf{1} \right\|_1 \tag{4.10}$$

to penalise the scaling parameters $\mathbf{s}_i$ (i.e. stretch of the ellipsoid) by its difference to the mean $\tilde{\mathbf{s}}_i$. As shown in Figure 4.3, this encourages sphericality and avoids the problem of Gaussians which are highly elongated along the viewing direction, creating artefacts. Let the union of the keyframes in the current window, and the randomly selected one be $\mathcal{W} = \mathcal{W}_k \cup \mathcal{W}_r$. For mapping, we solve the following problem:

$$\min_{\substack{\mathbf{T}_{CW}^k \in \mathbf{SE}(3), \mathcal{G}, \\ \forall k \in \mathcal{W}}} \sum_{\forall k \in \mathcal{W}} E_{pho}^k + \lambda_{iso} E_{iso} \ . \tag{4.11}$$

If depth observations are available, as in tracking, geometric residuals Equation 4.8 are added to the optimisation problem:

$$\min_{\substack{\mathbf{T}_{CW}^k \in \mathbf{SE}(3), \mathcal{G}, \\ \forall k \in \mathcal{W}}} \sum_{\forall k \in \mathcal{W}} (\lambda_{pho} E_{pho}^k + (1 - \lambda_{pho}) E_{geo}^k) + \lambda_{iso} E_{iso} \ .$$

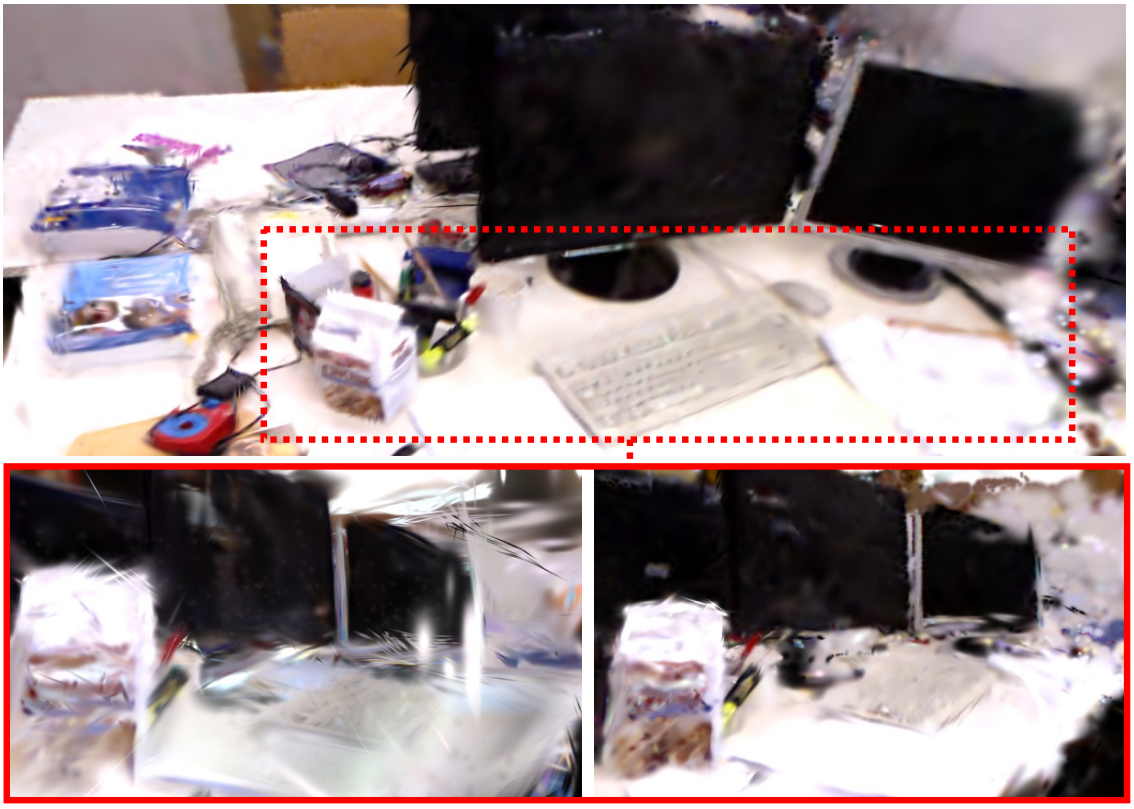


Figure 4.3: **Effect of isotropic regularisation**: **Top:** Rendering close to a training view (looking at the keyboard). **Bottom:** Rendering 3D Gaussians far from the training views (view from a side of the keyboard) without (left) and with (right) the isotropic loss. When the photometric constraints are insufficient, the Gaussians tend to elongate along the viewing direction, creating artefacts in the novel views and affecting the camera tracking.

We set $\lambda_{pho} = 0.9$ for all RGB-D experiments, and $\lambda_{iso} = 10$ for both monocular and RGB-D experiments.

4.4 Evaluation

We conduct a comprehensive evaluation of our system across a range of both real and synthetic datasets. Additionally, we perform an ablation study to justify our design choices. Finally, we present qualitative results of our system operating live using a monocular camera, illustrating its practicality and high-fidelity reconstruction.

4.4.1 Datasets

For our quantitative analysis, we evaluate our method on the TUM RGB-D dataset [Sturm et al., 2012] (3 sequences) and the Replica dataset [Straub et al., 2019] (8 sequences), following the evaluation in [Sucar et al., 2021]. For qualitative results, we use self-captured real-world sequences recorded by Intel Realsense d455. Since the Replica dataset is designed for RGB-D SLAM evaluation, it contains challenging purely rotational camera motions. Hence, we use the Replica dataset

only for RGB-D evaluation. The TUM RGB-D dataset is used for both monocular and RGB-D evaluation.

4.4.2 Implementation Details

We run our SLAM on a desktop with Intel Core i9 12900K 3.50GHz and a single NVIDIA GeForce RTX 4090. We present results from our multi-process implementation aimed at live, near real-time[1] applications. For a fair comparison with other methods on Replica, we additionally report results for single-process implementation, which performs more mapping iterations. As with 3DGS, time-critical rasterisation and gradient computation are implemented using CUDA. The rest of the SLAM pipeline is developed with PyTorch.

We use the Adam Optimiser for both camera poses and Gaussian parameter optimisation. For camera poses, we used 0.003 for rotation and 0.001 for translation. For 3D Gaussians, we used the default learning parameters of the original Gaussian Splatting implementation [Kerbl et al., 2023], apart from in monocular setting where we increase the learning rate of the positions of the Gaussians $\boldsymbol{\mu}_W$ by a factor of 10.

4.4.3 Metrics

For camera tracking accuracy, we report the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE) of the keyframes. To evaluate map quality, we report standard photometric rendering quality metrics (PSNR, SSIM and LPIPS) following the evaluation protocol used in [Sandström et al., 2023]. To evaluate the map quality, rendering metrics are computed on every fifth frame. We exclude the keyframes (training views). We report the average across three runs for all our evaluations. In the tables, the best result is in bold, and the second best is underlined.

4.4.4 Baseline Methods

We primarily benchmark our SLAM method against other approaches that, like ours, do not have explicit loop closure. In monocular settings, we compare with state-of-the-art classical and learning-based direct visual odometry (VO) methods. Specifically, we compare DSO [Engel et al., 2017], DepthCov [Dexheimer and Davison, 2023], and DROID-SLAM [Teed and Deng, 2021] in VO configurations. These methods are selected based on their public reporting of results on the benchmark (TUM dataset) or the availability of their source code for getting the benchmark result. In the RGB-D case, we compare against neural-implicit SLAM methods [Sucar et al., 2021, Zhu et al., 2022, Huang et al., 2021, Yang et al., 2022, Johari et al., 2023, Wang et al., 2023, Sandström et al., 2023] which are also map-centric, rendering-based and do not perform loop closure.

[1] By real-time performance, we mean that the tracking is performed at frame rate (e.g., 30 FPS)

Input	Loop-closure	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Monocular	w/o	DSO [Engel et al., 2017]	22.4	**1.10**	9.50	11.0
		DROID-VO [Teed and Deng, 2021]	5.20	10.7	7.30	7.73
		DepthCov-VO [Dexheimer and Davison, 2023]	5.60	1.20	68.8	25.2
		Ours	**3.78**	4.60	**3.50**	**3.96**
	w/	DROID-SLAM [Teed and Deng, 2021]	**1.80**	**0.50**	2.80	1.70
		ORB-SLAM2 [Mur-Artal and Tardós, 2017]	1.90	0.60	**2.40**	**1.60**
RGB-D	w/o	iMAP [Sucar et al., 2021]	4.90	2.00	5.80	4.23
		NICE-SLAM [Zhu et al., 2022]	4.26	6.19	3.87	4.77
		DI-Fusion [Huang et al., 2021]	4.40	2.00	5.80	4.07
		Vox-Fusion [Yang et al., 2022]	3.52	1.49	26.01	10.34
		ESLAM [Johari et al., 2023]	2.47	**1.11**	2.42	2.00
		Co-SLAM [Wang et al., 2023]	2.40	1.70	2.40	2.17
		Point-SLAM [Sandström et al., 2023]	4.34	1.31	3.48	3.04
		Ours	**1.50**	1.44	**1.49**	**1.47**
	w/	BAD-SLAM [Schöps et al., 2019]	1.70	1.10	1.70	1.50
		Kintinous [Whelan et al., 2015a]	3.70	2.90	3.00	3.20
		ORB-SLAM2 [Mur-Artal and Tardós, 2017]	**1.60**	**0.40**	**1.00**	**1.00**

Table 4.1: **Camera tracking result on TUM for monocular and RGB-D.** ATE RMSE in cm is reported. In both monocular and RGB-D cases, we achieve state-of-the-art performance. In particular, in the monocular case, not only do we outperform systems which use deep prior, but we achieve comparable performance with many of the RGB-D systems.

4.4.5 Camera Tracking Accuracy

Table 4.1 shows the tracking results on the TUM RGB-D dataset. In the monocular setting, our method surpasses other baselines without requiring any deep priors. Furthermore, our performance is comparable to systems which perform explicit loop closure. This clearly highlights that there still remains potential for enhancing the tracking of monocular SLAM by exploring fundamental SLAM representations.

Our RGB-D method shows better performance than any other baseline method. Notably, our system surpasses ORB-SLAM in the fr1/desk sequence, narrowing the gap between Map-centric SLAM and the state-of-the-art sparse frame-centric methods. Table 4.2 reports results on the synthetic Replica dataset. Our single-process implementation shows competitive performance and achieves the best result in 6 out of 8 sequences. Our multi-process implementation which performs fewer mapping iterations, still performs comparably. In contrast to other methods, our system demonstrates higher performance on real-world data (TUM RGB-D) as the Gaussian positions are optimised to compensate for the sensor noise.

4.4.6 Novel View Rendering

Table 4.6 summarises the novel view rendering performance of our method with RGB-D input. We consistently show the best performance across most sequences and are at least second best.

Method	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
iMAP [Sucar et al., 2021]	3.12	2.54	2.31	1.69	1.03	3.99	4.05	1.93	2.58
NICE-SLAM [Zhu et al., 2022]	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.07
Vox-Fusion [Yang et al., 2022]	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
ESLAM [Johari et al., 2023]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	**0.63**	0.63
Point-SLAM [Sandström et al., 2023]	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72	0.53
Ours	0.44	0.32	0.31	0.44	0.52	**0.23**	0.17	2.25	0.58
Ours (sp)	**0.33**	**0.22**	**0.29**	**0.36**	**0.19**	0.25	**0.12**	0.81	**0.32**

Table 4.2: **Camera tracking result on Replica for RGB-D SLAM.** ATE RMSE in cm is reported. We achieve best performance across most sequences. Here, Ours is our multi-process implementation and Ours (sp) is the single-process implementation which ensures a certain amount of mapping iteration similar to other works.

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Mono	w/o E_{iso}	4.16	4.66	5.73	4.83
	w/o kf selection	13.2	**4.36**	8.65	8.73
	w/o pruning	78.2	4.5	57.0	46.6
	Ours	**3.78**	4.60	**3.50**	**3.96**
RGB-D	w/o E_{geo}	2.39	**0.62**	4.98	2.66
	w/o kf selection	1.64	1.49	2.60	1.90
	w/o E_{iso}	1.60	**1.42**	**1.32**	**1.43**
	Ours	**1.50**	1.44	1.49	1.47

Table 4.3: **Ablation Study on TUM RGB-D dataset.** We analyse the usefulness of isotropic regularisation, geometric residual, keyframe selection and pruning to our SLAM system.

Method	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
w/o E_{iso}	**0.44**	0.86	**0.28**	0.75	0.99	0.36	0.28	2.6	0.82
Ours	**0.44**	**0.32**	0.31	**0.44**	**0.52**	**0.23**	**0.17**	**2.25**	**0.58**

Table 4.4: **Isotropic Loss Ablation Study on Replica dataset (RGB-D input).** Numbers are camera tracking error (ATE RMSE) in cm.

Memory Usage [MB]				
iMAP [Sucar et al., 2021]	NICE-SLAM [Zhu et al., 2022]	Co-SLAM [Wang et al., 2023]	Ours (Mono)	Ours (RGB-D)
0.8MB	40.3.4MB	6.4MB	2.6MB	3.97MB

Table 4.5: **Memory Analysis on TUM RGB-D dataset.** The baseline numbers are computed from the parameter numbers in [Wang et al., 2023]

Our rendering FPS is hundreds of times faster than other methods, offering a significant advantage for applications which require real-time map interaction. While Point-SLAM is competitive, the method focuses on view synthesis rather than novel-view synthesis. Their view synthesis is conditional on the availability of depth due to the depth-guided ray-sampling, making novel-view synthesis challenging. On the other hand, our rasterisation-based approach does not require depth guidance and achieves efficient, high-quality, novel view synthesis. Figure 4.4 provides a qualitative comparison of the rendering of ours and Point-SLAM (with depth guidance).

Method	Metric	room0	room1	room2	office0	office1	office2	office3	office4	Avg.	Rendering FPS
NICE-SLAM [Zhu et al., 2022]	PSNR[dB] ↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42	0.54
	SSIM ↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809	
	LPIPS↓	0.33	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233	
Vox-Fusion [Yang et al., 2022]	PSNR[dB] ↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41	2.17
	SSIM ↑	0.683	0.751	0.798	0.857	0.876	0.794	0.803	0.847	0.801	
	LPIPS↓	0.303	0.269	0.234	0.241	0.184	0.243	0.213	0.199	0.236	
Point-SLAM [Sandström et al., 2023]	PSNR[dB] ↑	32.40	34.08	35.5	38.26	39.16	33.99	33.48	33.49	35.17	1.33
	SSIM ↑	**0.974**	**0.977**	**0.982**	**0.983**	**0.986**	0.96	0.960	**0.979**	**0.975**	
	LPIPS↓	0.113	0.116	0.111	0.1	0.118	0.156	0.132	0.142	0.124	
Ours	PSNR[dB] ↑	**34.83**	**36.43**	**37.49**	**39.95**	**42.09**	**36.24**	**36.7**	**36.07**	**37.50**	**769**
	SSIM ↑	0.954	0.959	0.965	0.971	0.977	**0.964**	**0.963**	0.957	0.960	
	LPIPS↓	**0.068**	**0.076**	**0.075**	**0.072**	**0.055**	**0.078**	**0.065**	**0.099**	**0.070**	

Table 4.6: **Rendering performance comparison of RGB-D SLAM methods on Replica.** Our method outperforms most of the rendering metrics compared to existing methods. Note that Point-SLAM uses sensor depth (ground-truth depth in Replica) to guide sampling along rays, which limits the rendering performance to existing views. The numbers for the baselines are taken from [Sandström et al., 2023].

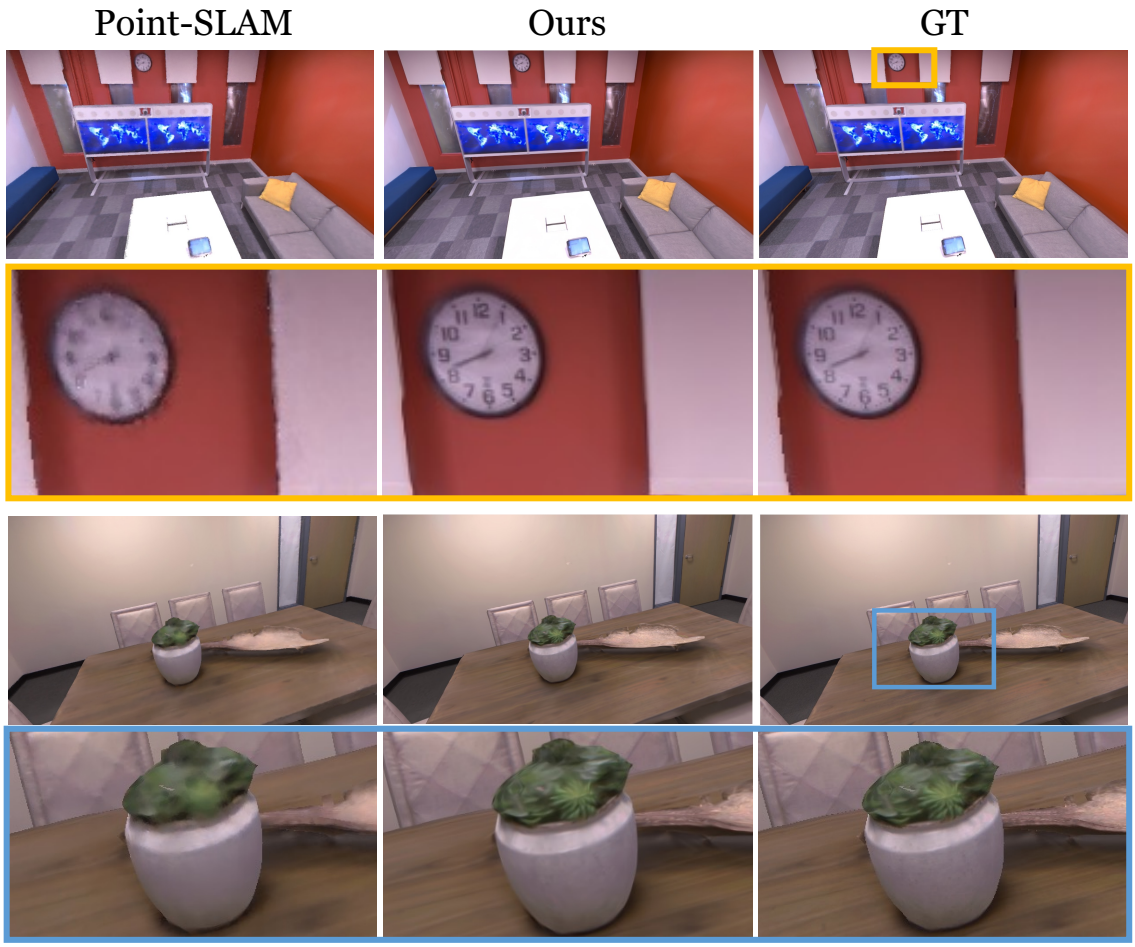


Figure 4.4: **Rendering examples on Replica.** Point-SLAM struggle with rendering fine details due to the stochastic ray sampling.

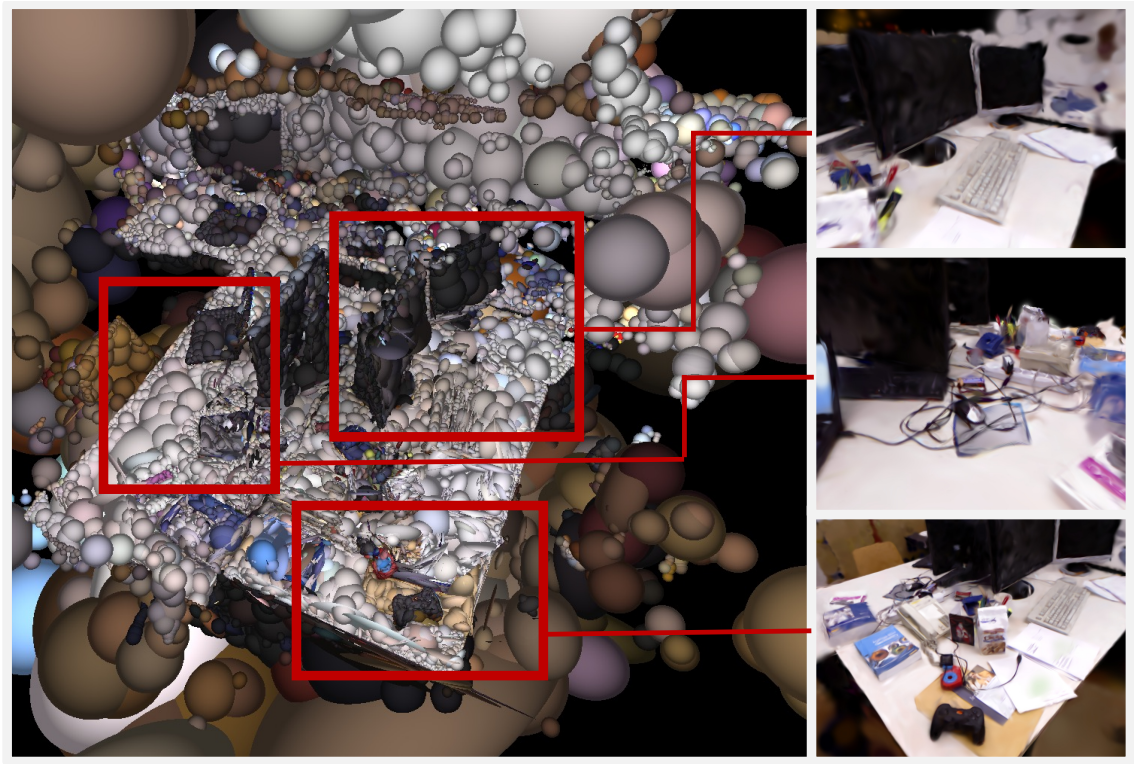


Figure 4.5: **Monocular SLAM result on fr1/desk sequence:** We show the reconstructed 3D Gaussian maps (Left) and novel view synthesis result (Right).

4.4.7 Ablative Analysis

In Table 4.3, we perform ablation to confirm our design choices. Isotropic regularisation and geometric residual improve the tracking of monocular and RGB-D SLAM respectively, as they aid in constraining the geometry when photometric signals are weak. For both cases, keyframe selection significantly improves systems performance, as it automatically chooses suitable keyframes based on our occlusion-aware keyframe selection and management. In TUM, for RGB-D SLAM, as Table 4.3 shows, isotropic regularisation does not improve the performance but only shows a marginal difference. However, for Replica, as summarised in Table 4.4, isotropic loss significantly improves camera tracking performance. Even with the depth measurement, since rasterisation does not consider the elongation along the viewing axis. Isotropic regularisation is required to prevent the Gaussians from over-stretching, especially for textureless regions, which are common in Replica. We further compare the memory usage of different 3D representations in Table 4.5. MLP-based iMAP is clearly more memory efficient, but it struggles to express high-fidelity 3D scenes due to the limited capacity of small MLP. Compared with a voxel grid of features used in NICE-SLAM, our method uses significantly less memory.

4.4.8 Qualitative Results

We report both the 3D reconstruction of the SLAM dataset and self-captured sequences. In Figure 4.5, we visualise the monocular SLAM reconstruction of fr1/desk. The placements of the

Figure 4.6: **Self-captured Scenes:** Challenging scenes and objects, for example, transparent glasses and the crinkled texture of salad, are captured by our monocular SLAM running live.

Gaussians are geometrically sensible and coherent in 3D, and our rendering from the different viewpoints highlights the quality of our systems' novel view synthesis. In Figure 4.6, we self-capture challenging scenes for monocular SLAM. By not explicitly modelling a surface, our system naturally handles transparent objects which is challenging for many other SLAM systems.

4.5 Conclusion

We have proposed the first SLAM method using 3D Gaussians as a SLAM representation. Via efficient volume rendering, our system significantly advances the fidelity and diversity of object materials a live SLAM system can capture. Our system achieves state-of-the-art performance across benchmarks for both monocular and RGB-D cases. Targetting photorealistic reconstruction, our system compresses the input images into a unified 3D representation that is much smaller than the original images in terms of memory (*e.g.* Replica/Office0 contains over 500 MB of images, and the reconstruction size is only 13 MB), and as shown in Figure 1.4, we can apply off-the-shelf pre-trained models to rendered images as our reconstruction retained, though not perfect, the details of the natural images.

One of the limitation of our approach is that our method requires relatively slow motion for monocular tracking since our system is slower than other state-of-the-art SLAM systems such as ORB-SLAM. Speeding up the tracking by preconditioning or approximate second-order op-

timiser would be an interesting and practical future direction.

CHAPTER 5

Gaussian Belief Propagation For State Estimation

Contents

In this section, we first introduce the technical derivation of Gaussian Belief Propagation (GBP) following [Davison and Ortiz, 2019] and detail how we can extend GBP to handle non-linearity and outlying measurements. We then extend GBP to support variables that belong to Lie Groups.

5.1 Gaussian Belief Propagation

GBP is a specialisation of Loopy Belief Propagation (LBP), where all variables, factors, and hence, all the inter-node messages are Gaussian. Here, all the Gaussians are parameterised in the canonical form unless otherwise specified.

5.1.1 Belief Propagation

Here, we reiterate the summary of the message-passing rules in Belief Propagation (BP) from Section 2.4 for convenience.

Belief Update The marginal distribution of a variable x_i is the product of all incoming messages:

$$p(x_i) = \prod_{\alpha \in n(x_i)} m_{f_\alpha \to x_i}(x_i) \, . \tag{5.1}$$

Variable-to-factor Message The message from the variable x_i to the factor f_α is:

$$m_{x_i \to f_\alpha}(x_i) = \prod_{\beta \in n(x_i)/\alpha} m_{f_\beta \to x_i}(x_i) \, . \tag{5.2}$$

Factor-to-variable Message The message from the factor f_α to the variable x_i is:

$$m_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{x}_\alpha / x_i} f_\alpha(\mathbf{x}_\alpha) \prod_{x_j \in \mathbf{x}_\alpha / x_i} m_{x_j \to f_\alpha}(x_j) \, . \tag{5.3}$$

5.1.2 Belief Update

Following Equation 5.1, belief update is a product of all the incoming Gaussians. A product of the Gaussians in a canonical form is a summation of the parameters (Equation 2.7). Hence, to update the belief of a variable node x_i, we simply take the summation of the parameters of all the incoming factor-to-variable messages:

$$\boldsymbol{\eta}_i = \sum_{f_\alpha \in n(x_i)} \boldsymbol{\eta}_{f_\alpha}, \quad \boldsymbol{\Lambda}_i = \sum_{f_\alpha \in n(x_i)} \boldsymbol{\Lambda}_{f_\alpha} \, , \tag{5.4}$$

and the updated belief is $p(x_i) = \mathcal{N}^{-1}(\mathbf{x}; \boldsymbol{\eta}_i, \boldsymbol{\Lambda}_i)$.

5.1.3 Variable to Factor Message

As shown in Equation 5.2, the message from a variable x_i to a factor f_α is the product of all the incoming factor-to-variable messages except the one from the outgoing factor:

$$\boldsymbol{\eta}_{x_i \to f_\alpha} = \sum_{f_\beta \in n(x_i)/f_\alpha} \boldsymbol{\eta}_{f_\beta}, \quad \boldsymbol{\Lambda}_{x_i \to f_\alpha} = \sum_{f_\beta \in n(x_i)/f_\alpha} \boldsymbol{\Lambda}_{f_\beta} \, , \tag{5.5}$$

where we define the incoming messages as $m_{f_\beta \to x_i}(x_i) = \mathcal{N}^{-1}(\boldsymbol{\eta}_{f_\beta}, \boldsymbol{\Lambda}_{f_\beta})$, and the outgoing message as $m_{x_i \to f_\alpha}(x_i) = \mathcal{N}^{-1}(\boldsymbol{\eta}_{x_i \to f_\alpha}, \boldsymbol{\Lambda}_{x_i \to f_\alpha})$.

We also can view the message as downdating the belief of x_i with the message from f_α:

$$m_{x_i \to f_\alpha}(x_i) = \frac{p(x_i)}{m_{f_\alpha \to x_i}(x_i)} = \frac{\prod_{f_\beta \in n(x_i)} m_{f_\beta \to x_i}(x_i)}{m_{f_\alpha \to x_i}(x_i)} \, , \tag{5.6}$$

which avoids recomputing the partial sum for every outgoing message.

5.1.4 Factor to Variable Message

Given a measurement function h, we can define a likelihood, the conditional probability of $\mathbf{z}$ given $\mathbf{x}$ as:

$$p(\mathbf{z}|\mathbf{x}) = K \exp\left(-\frac{1}{2}(\mathbf{z} - h(\mathbf{x}))^\top \mathbf{\Sigma}^{-1}(\mathbf{z} - h(\mathbf{x}))\right) , \tag{5.7}$$

$$= K \exp\left(-E(\mathbf{x}; \mathbf{z})\right) , \tag{5.8}$$

where K is the normalisation constant, and $\mathbf{\Sigma}$ is the covariance matrix of the measurement noise, and $\mathbf{z}$ is the measurement. We define an energy function $E(\mathbf{x}; \mathbf{z})$ which is proportional to the negative log of the likelihood.

To derive the Gaussian factor over $\mathbf{x}$, we begin with our general factor energy $E(\mathbf{x}; \mathbf{z})$, and our goal is to manipulate the energy into the form of a Gaussian function over $\mathbf{x}$ in the canonical form:

$$E(\mathbf{x}; \mathbf{z}) = \frac{1}{2}\mathbf{x}^\top \mathbf{\Lambda}\mathbf{x} - \boldsymbol{\eta}^\top \mathbf{x} . \tag{5.9}$$

After transforming the energy into this form, we can identify the canonical parameters $\boldsymbol{\eta}$ and $\mathbf{\Lambda}$ of the factor distribution.

To begin with, any linear measurement function can be generally written as:

$$h(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} , \tag{5.10}$$

where $\mathbf{b} \in \mathbb{R}^m$ is a constant vector and $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a constant matrix for $\mathbf{z} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$.

By substituting Equation 5.10 into Equation 5.7, we can write the energy function as:

$$E(\mathbf{x}; \mathbf{z}) = \frac{1}{2}(\mathbf{z} - \mathbf{A}\mathbf{x} - \mathbf{b})^\top \mathbf{\Lambda}(\mathbf{z} - \mathbf{A}\mathbf{x} - \mathbf{b}) , \tag{5.11}$$

where $\mathbf{\Lambda} = \mathbf{\Sigma}^{-1}$. Re-arranging the terms, we can write the energy function as:

$$\begin{aligned} E(\mathbf{x}; \mathbf{z}) &= \frac{1}{2}\left[\mathbf{z} - \mathbf{A}\mathbf{x} - \mathbf{b}\right]^\top \mathbf{\Lambda}\left[\mathbf{z} - \mathbf{A}\mathbf{x} - \mathbf{b}\right] \\ &= \frac{1}{2}\left[(\mathbf{z} - \mathbf{b}) - \mathbf{A}\mathbf{x}\right]^\top \mathbf{\Lambda}\left[(\mathbf{z} - \mathbf{b}) - \mathbf{A}\mathbf{x}\right] \\ &= \frac{1}{2}\Big[(\mathbf{z} - \mathbf{b})^\top \mathbf{\Lambda}(\mathbf{z} - \mathbf{b}) + (\mathbf{A}\mathbf{x})^\top \mathbf{\Lambda}\mathbf{A}\mathbf{x} - (\mathbf{z} - \mathbf{b})^\top \mathbf{\Lambda}\mathbf{A}\mathbf{x} - (\mathbf{A}\mathbf{x})^\top \mathbf{\Lambda}(\mathbf{z} - \mathbf{b})\Big] . \end{aligned}$$

The first of the four terms here is a constant which doesn't depend on $\mathbf{x}$, so we can drop it into the normalising constant c. The third and fourth are equal (one is the transpose of the other, and both are scalars), so we can simplify to:

$$\begin{aligned} E(\mathbf{x}; \mathbf{z}) &= \frac{1}{2}(\mathbf{A}\mathbf{x})^\top \mathbf{\Lambda}\mathbf{A}\mathbf{x} - (\mathbf{z} - \mathbf{b})^\top \mathbf{\Lambda}\mathbf{A}\mathbf{x} + c \\ &= \frac{1}{2}\mathbf{x}^\top (\mathbf{A}^\top \mathbf{\Lambda}\mathbf{A})\mathbf{x} - \left(\mathbf{A}^\top \mathbf{\Lambda}(\mathbf{z} - \mathbf{b})\right)^\top \mathbf{x} + c . \end{aligned} \tag{5.12}$$

Matching this with the Equation 5.9, we can identify the canonical form parameters of the factor as:

$$\boldsymbol{\eta}_f = \mathbf{A}^\top \mathbf{\Lambda}(\mathbf{z} - \mathbf{b}), \tag{5.13}$$

$$\mathbf{\Lambda}_f = \mathbf{A}^\top \mathbf{\Lambda}\mathbf{A} . \tag{5.14}$$

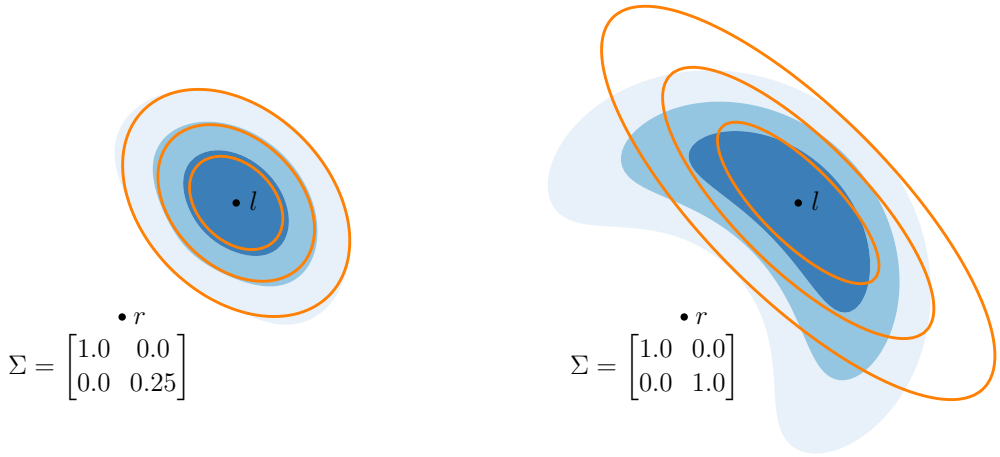


Figure 5.1: Linear approximation (orange) of the true non-linear range-bearing factor (blue). Figure adapted from [Ortiz et al., 2021].

The constant c can be ignored as it will be absorbed into the normalisation constant K.

Following Equation 5.3, the message from the factor f_α to the variable x_i is the marginal of the joint distribution of the factor f_α and all the incoming messages except from x_i. Again, the joint distribution is computed by the product, which simply is the summation of the parameters. Marginalisation in a canonical form involves the Schurs complement of the joint Gaussian (Equation 2.12); however, typically, since the factors are only adjacent to a small number of variables, the operation is efficient.

5.2 Beyond Gaussian Factors

In any realistic problem, the Gaussian assumption about the sensor measurement may not hold as a sensor model might be non-linear or observations may contain outlying noise. Here, we discuss how GBP can be extended to support the *non-linearity* of the measurement function and also how we can *robustify* the inference against outlying measurement.

5.2.1 Non-linear Factors

We have so far assumed that the factors are linear; however, most problems involve factors with non-linearity in their measurement prediction function $h(\cdot)$, which means that the factor distribution over the variables $\mathbf{x}$ is no longer Gaussian. The common practice is to linearise the problem by local linear approximation of the functions, and following [Davison and Ortiz, 2019], we apply the same approach for GBP. First, we take the Taylor approximation of the measurement

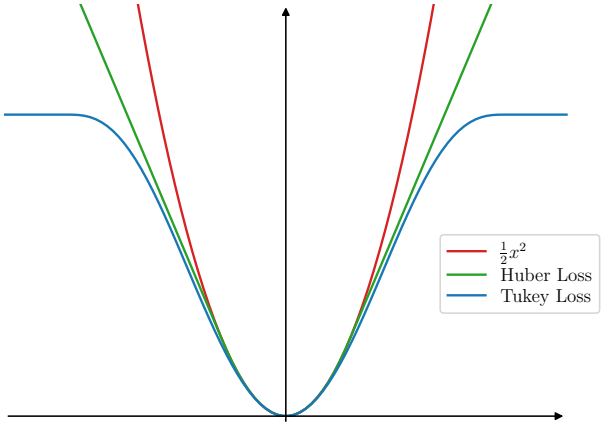


Figure 5.2: Huber and Tukey loss plotted against a quadratic loss. We set the parameters to the commonly used value of $c = 1.345, 4.685$, respectively.

prediction function around the current state $\mathbf{x}_t$:

$$h(\mathbf{x}) \approx h(\mathbf{x}_t) + \nabla h(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}^{\top}(\mathbf{x} - \mathbf{x}_t) \,. \tag{5.15}$$

Let $\mathbf{J} = \nabla h(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}^{\top}$ be the Jacobian of the measurement prediction function evaluated at the current state $\mathbf{x}_t$, them the linearised function can be written as:

$$h(\mathbf{x}) \approx \mathbf{J}\mathbf{x} + (h(\mathbf{x}_t) - \mathbf{J}\mathbf{x}_t) \,. \tag{5.16}$$

By comparing against Equation 5.10, we identify that $\mathbf{A} = \mathbf{J}$ and $\mathbf{b} = h(\mathbf{x}_t) - \mathbf{J}(\mathbf{x}_t)$. Substituting them into Equation 5.13 and Equation 5.14, we get the canonical parameters of a linearised approximation of a non-linear factor to be:

$$\boldsymbol{\eta}_f = \mathbf{J}^{\top}\boldsymbol{\Lambda}(\mathbf{z} - h(\mathbf{x}_t) + \mathbf{J}\mathbf{x}_t), \tag{5.17}$$

$$\boldsymbol{\Lambda}_f = \mathbf{J}^{\top}\boldsymbol{\Lambda}\mathbf{J} \,. \tag{5.18}$$

For example, we use Figure 5.1 to visualise the linear approximation of the range-bearing factor, which is defined as:

$$h(\mathbf{x}) = \begin{bmatrix} \|r - l\|_2 \\ \arctan(l_y - r_y, l_x - r_x) \end{bmatrix} \,, \tag{5.19}$$

where r is the robot pose and l is the landmark pose. The linear approximation is in orange, and the true non-linear factor is in blue. As shown, the approximation becomes worse as we increase the measurement uncertainty.

5.2.2 Robust Factors

Practically, for any problem, we have a certain ratio of outlying measurements which does not follow the assumed Gaussian distribution. Outlying observations yield a high residual due to the quadratic cost. Thus, the presence of even a single outlier can cause optimisation to fail and produce a distorted solution.

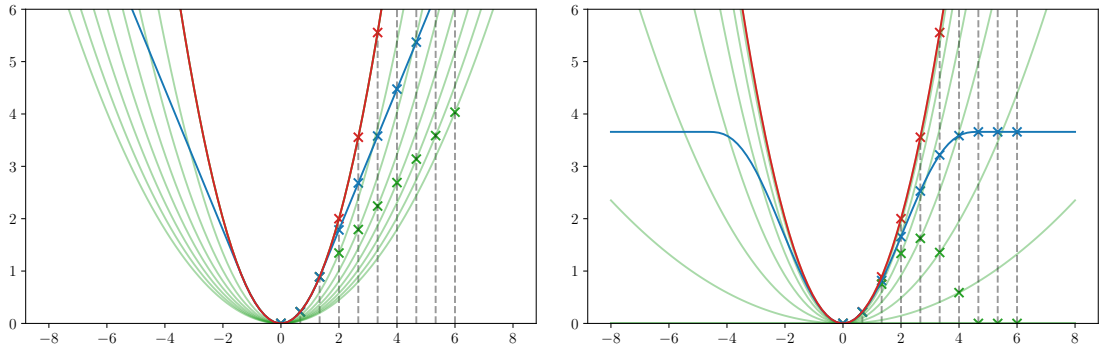


Figure 5.3: We visualise how the parabola is scaled to match the gradient of the robust cost function. **Left:** Huber Loss, **Right:** Tukey Loss.

This problem is encountered commonly in a non-linear least-squares problem. Here, we will apply the same strategy to mitigate the outlying measurements and extend the GBP algorithm to handle such cases.

First, let us revisit the standard non-linear least-squares problem:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{z}_i - h_i(\mathbf{x})\|^2_{\mathbf{\Sigma}} \,, \tag{5.20}$$

where $h(\cdot)$ is a non-linear measurement function. As discussed in Section 2.2.4, algorithms such as Gauss-Newton avoid the costly computation of the Hessian by exploiting the objective function's least-squares structure. In order to down-weight the influence of the outlying measurement while retaining the least-squares structure, we add a scalar weighting function $w_i(\cdot)$:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} w_i(\mathbf{x}_t) \|\mathbf{z}_i - h_i(\mathbf{x})\|^2_{\mathbf{\Sigma}} \,, \tag{5.21}$$

which is computed based on the current state $\mathbf{x}_t$ for each of the energies. This form of the problem is called **Iteratively Reweighted Least Squares** and can be interpreted as dynamically scaling per measurement covariance as:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} w_i(\mathbf{x}_t) \|\mathbf{z}_i - h_i(\mathbf{x})\|^2_{\mathbf{\Sigma}} \,, \tag{5.22}$$

$$= \arg\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} w_i(\mathbf{x}_t) \left(\mathbf{z}_i - h_i(\mathbf{x})\right)^\top \mathbf{\Sigma}_i^{-1} \left(\mathbf{z}_i - h_i(\mathbf{x})\right) \,, \tag{5.23}$$

$$= \arg\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} \left(\mathbf{z}_i - h_i(\mathbf{x})\right)^\top \left(w_i(\mathbf{x}_t) \mathbf{\Sigma}_i^{-1}\right) \left(\mathbf{z}_i - h_i(\mathbf{x})\right) \,, \tag{5.24}$$

since $w_i(\mathbf{x})$ is a scalar. Since we are simply re-scaling the covariance, the problem structure remains unchanged, and we can use efficient algorithms such as Gauss-Newton as Levenberg-Marquardt.

Let now discuss how we can find a *good* weighting function. One way to reduce the effect of the outlying measurements is to avoid quadratic costs. Hence, by switching from a quadratic

to a linear cost after a certain threshold c, we can reduce the overall influence of the outlying measurements. Such loss function is called the *Huber Loss*, as shown in Figure 5.2, which is a piecewise cost function which assigns less energy to the outlying measurements. *Tukey's Biweight* on the other hand clips the cost at a certain value and practically ignores the measurements in the outlying region during optimisation. Both Huber Loss and Tukey's Biweight belong to *M-estimators*, a robust alternative to the quadratic cost [Zhang, 1997].

To replace the standard quadratic cost function with a robust function $\rho(\cdot)$, we refine the energy as:

$$E(\mathbf{x}; \mathbf{z}) = \rho\left(\|\mathbf{z} - h(\mathbf{x})\|_{\mathbf{\Sigma}}\right) \ . \tag{5.25}$$

Note, we get the standard least-square by setting $\rho(u) = \frac{1}{2}u^2$.

The definitions of commonly used M-estimators are:

Huber Loss:

$$\rho(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq c \\ c(|x| - \frac{c}{2}) & \text{otherwise} \end{cases} \tag{5.26}$$

Tukey's Biweight:

$$\rho(x) = \begin{cases} \frac{c^2}{6}\left(1 - \left(1 - \left(\frac{x}{c}\right)^2\right)^3\right) & \text{if } |x| \leq c \\ \frac{c^2}{6} & \text{otherwise} \end{cases} \tag{5.27}$$

Notice that we cannot simply replace $\rho(\cdot)$ from a quadratic function to Huber as they will no longer have the least-squares form. Instead of solving the problem directly, we derive appropriate weighting functions and solve it using iteratively reweighted least-squares.

Let define a general *robust* optimisation problem as:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{n} \rho\left(\|\mathbf{z}_i - h_i(\mathbf{x})\|_{\mathbf{\Sigma}}\right) \ , \tag{5.28}$$

$$= \arg\min_{\mathbf{x}} \sum_{i=1}^{n} \rho\left(r_i(\mathbf{x})\right) \ . \tag{5.29}$$

Taking the partial derivative with respect to $\mathbf{x}$:

$$\nabla_{\mathbf{x}} \sum_{i=1}^{n} \rho\left(r_i(\mathbf{x})\right) = \sum_{i=1}^{n} \rho'\left(r_i(\mathbf{x})\right) \nabla_{\mathbf{x}} r_i(\mathbf{x}) = 0 \ , \tag{5.30}$$

where $\rho'\left(r_i(\mathbf{x})\right) = \frac{\partial \rho(r_i(\mathbf{x}))}{\partial r_i(\mathbf{x})}$ and is referred to as an influence function $\psi(\cdot)$:

$$\nabla_{\mathbf{x}} \sum_{i=1}^{n} \rho\left(r_i(\mathbf{x})\right) = \sum_{i=1}^{n} \psi\left(r_i(\mathbf{x})\right) \nabla_{\mathbf{x}} r_i(\mathbf{x}) = 0 \ . \tag{5.31}$$

We now want to find a weighting function $w(\cdot)$ which matches against the above robust optimisation problem. Taking the partial derivative of Equation 5.21:

$$\nabla_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} w_i(\mathbf{x}_t) \|\mathbf{z}_i - h_i(\mathbf{x})\|^2_{\mathbf{\Sigma}} = \nabla_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} w_i(\mathbf{x}) r_i(\mathbf{x})^2 = 0 \,, \tag{5.32}$$

$$= \sum_{i=1}^{n} w_i(\mathbf{x}) r_i(\mathbf{x}) \nabla_{\mathbf{x}} r_i(\mathbf{x}) \,. \tag{5.33}$$

Equating Equation 5.31 to Equation 5.33, we get:

$$w_i(\mathbf{x}) r_i(\mathbf{x}) = \psi\left(r_i(\mathbf{x})\right) \,, \tag{5.34}$$

and by rearranging:

$$w_i(\mathbf{x}) = \frac{\psi\left(r_i(\mathbf{x})\right)}{r_i(\mathbf{x})} \,. \tag{5.35}$$

Solving iteratively reweighted least-squares problems with such a weighting function yields the same solution as solving the robustified problem (as the gradient is the same at every step of the optimisation problem), and we still get to exploit the problem structure and use algorithms such as Gauss-Newton / Levenberg-Marquardt.

Returning to GBP, we can simply robustify the factors by applying the same modifications. Following Equation 5.24, for each factor energy, we dynamically re-scale the inverse measurement covariance, thus the measurement precision. With this, we define the canonical parameter of the factors of Iteratively Reweighted GBP as:

$$\boldsymbol{\eta}_f = w_i(\mathbf{x}_t) \cdot \mathbf{J}^\top \mathbf{\Lambda} (\mathbf{z} - h(\mathbf{x}_t) + \mathbf{J}\mathbf{x}_t), \tag{5.36}$$

$$\mathbf{\Lambda}_f = w_i(\mathbf{x}_t) \cdot \mathbf{J}^\top \mathbf{\Lambda} \mathbf{J} \tag{5.37}$$

5.3 Gaussian Belief Propagation with Lie Groups

The use of Lie theory is a key component of modern state estimation. So far, we have assumed that all of the variables are in Euclidean space; however, such an assumption does not hold in most robotics / 3D vision applications as a state of a pose is represented using rotation and translation, where careful thought about parameterisation is needed.

Here, we detail how we extend GBP to support inference of variables which belong to Lie Groups, and although optimisation on the manifold is well studied [Absil et al., 2008, Barfoot, 2017, Lynch and Park, 2017] and is used in many numerical optimisation libraries [Agarwal et al., 2010, Kümmerle et al., 2011, Dellaert, 2012], application of these machinaries to GBP is novel.

The core of how we use Lie Theory within GBP is the choice that *all messages to or from a Lie Group variables take the form of a point estimate, represented by the full over-parameterised Group element, together with a minimal precision matrix defined in the tangent space around that element*. So, when variable $\mathcal{X}$ represents a transformation which is a member of a Lie Group, all

messages to it will take the form $\mathcal{N}(\bar{\mathcal{X}}, \mathbf{\Lambda}^{-1})$, where $\mathbf{\Lambda}$ is a precision matrix in the tangent space at the point-estimate $\bar{\mathcal{X}}$.

This approach gives maximum flexibility and minimises the need for independent nodes to have knowledge or memory of each other (which might be required with alternative ideas, such as that messages would represent perturbations around some remembered linearisation point).

5.3.1 Uncertainty on the Manifold

First, we discuss how the uncertainty on the manifold is defined. Rotations and poses can be described as an element in a Lie group free from singularities but with some constraints such as orthonormality. Alternatively, it can be represented as an element in Lie algebra, which can be treated as vectors (since they are isomorphic to elements in a vector space); however, we must worry about singularities as we move far away from the origin of the tangent space. When expressing uncertainties of rotations and rigid transforms, we are motivated to exploit the vector space characteristics of Lie algebra, as we can reuse all the tools we commonly use in probabilistic methods [Barfoot, 2017, Chapter 7].

A random variable for elements on the manifold such as $\mathbf{SO}(3)$ can be defined in many different ways but in this thesis, we use the right perturbation convention, following the tutorial by [Solà et al., 2018]. For $\mathbf{SO}(3)$, the random variable $\mathcal{X}$ is defined as:

$$\mathcal{X} = \bar{\mathcal{X}} \oplus \xi \,, \tag{5.38}$$

where $\bar{\mathcal{X}} \in \mathbf{SO}(3)$ is a noise-free *mean* rotation and $\xi \in \mathfrak{so}(3)$ is a small noisy component (i.e., a regular random variable in a vector space). If we assume that the random variable follows a Gaussian noise, we define the random variable as $\xi \sim \mathcal{N}(0, \mathbf{\Sigma}_{\mathcal{X}})$. Importantly, even though we write $\mathbf{\Sigma}_{\mathcal{X}}$, the covariance is defined for the perturbation vector ξ, and such covariance is well defined as the dimension of the perturbation vector ξ matches the Degrees of Freedom (DoF) of $\mathcal{M}$, whereas the dimension of $\mathcal{X}$ is greater than the DoF for any non-trivial manifolds, hence the covariance will be ill-defined [1]. For the rest of the thesis, we use the notation $\mathcal{N}(\mathcal{X}; \bar{\mathcal{X}}, \mathbf{\Sigma}_{\mathcal{X}})$ to denote a random variable on a manifold.

5.3.2 Linearising Factors

Consider a measurement function $h(\cdot)$ which is a function of a Lie group element $\bar{\mathcal{X}} \in \mathcal{M}$. To linearise such a function, same as Equation 5.15 we perform Taylor expansion but we define the

[1] For example, if we choose quaternion parameterisation for $\bar{\mathcal{X}} \in \mathbf{SO}(3)$, the covariance defined on the perturbation vector is well-defined as $\dim(\xi) = 3$ and we get 3×3 covariance matrix which matches the $\mathrm{dof}(\bar{\mathcal{X}}) = 3$. Whereas, if we've defined the covariance on the group element, we get 4×4 covariance matrix since $\dim(\bar{\mathcal{X}}) = 4$ which is greater than degrees of freedom, making the covariance matrix ill-defined.

small perturbation vector $\bar{\boldsymbol{\tau}} \in \mathcal{T}_{\bar{\mathcal{X}}_t}\mathcal{M}$ to be on a tangent plane of $\bar{\mathcal{X}}_t$:

$$h(\bar{\mathcal{X}}) \approx h(\bar{\mathcal{X}}_t) + \nabla h(\bar{\mathcal{X}})|^{\top}_{\bar{\mathcal{X}}=\bar{\mathcal{X}}_t}(\bar{\mathcal{X}} \ominus \bar{\mathcal{X}}_t) \,, \tag{5.39}$$

$$= h(\bar{\mathcal{X}}_t) + \mathbf{J}\bar{\boldsymbol{\tau}} \,. \tag{5.40}$$

In Section 5.2.1, we compared the linearised $h(\mathbf{x})$ against $h(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ to identify the canonical parameters of the factor potential $f(\mathbf{x}; \mathbf{z}, \mathbf{x}_t) \propto \mathcal{N}^{-1}(\mathbf{x}; \boldsymbol{\eta}_f, \boldsymbol{\Lambda}_f)$. Here, repeating the same process yields a factor potential which is a function of $\boldsymbol{\tau}$, where $f(\bar{\mathcal{X}}; \mathbf{z}, \bar{\mathcal{X}}_t) \propto \mathcal{N}^{-1}(\boldsymbol{\tau}; \boldsymbol{\eta}_f, \boldsymbol{\Lambda}_f)$:

$$\boldsymbol{\eta}_f = \mathbf{J}^{\top}\boldsymbol{\Lambda}(\mathbf{z} - h(\bar{\mathcal{X}}_t)), \tag{5.41}$$

$$\boldsymbol{\Lambda}_f = \mathbf{J}^{\top}\boldsymbol{\Lambda}\mathbf{J} \,. \tag{5.42}$$

where we've identified $\mathbf{A} = \mathbf{J}$ and $\mathbf{b} = h(\bar{\mathcal{X}}_t)$. It is important to note that the factor potential is a function of $\boldsymbol{\tau}$, and we are finding an increment which minimises the energy:

$$E(\bar{\boldsymbol{\tau}}; \mathbf{z}, \bar{\mathcal{X}}_t) = \frac{1}{2}\bar{\boldsymbol{\tau}}^{\top}\boldsymbol{\Lambda}_f\bar{\boldsymbol{\tau}} - \boldsymbol{\eta}_f^{\top}\bar{\boldsymbol{\tau}} \,, \tag{5.43}$$

and not directly optimising the variable $\mathcal{X}$. Computing the increment and applying them via retractions makes sure that $\bar{\mathcal{X}}$ always stays on the manifold.

The above calculation can be applied to a composite manifold; for example, let a factor be connected to one variable in $\mathbb{R}^2$ and two variables in $\mathcal{M}$.:

$$\bar{\mathcal{X}} = \begin{pmatrix} \bar{\mathcal{X}}_1 \\ \bar{\mathcal{X}}_2 \\ \bar{\mathcal{X}}_3 \end{pmatrix} \in \langle \mathbb{R}^2, \mathcal{M}, \mathcal{M} \rangle \,, \tag{5.44}$$

then the factor potential will be formed on the composite tangent space:

$$\bar{\boldsymbol{\tau}} = \begin{pmatrix} \bar{\boldsymbol{\tau}}_1 \\ \bar{\boldsymbol{\tau}}_2 \\ \bar{\boldsymbol{\tau}}_3 \end{pmatrix} \in \langle \mathbb{R}^2, T_{\bar{\mathcal{X}}_2}\mathcal{M}, T_{\bar{\mathcal{X}}_3}\mathcal{M} \rangle \,, \tag{5.45}$$

where the increments $\bar{\boldsymbol{\tau}}$ are defined in Euclidian space or around the tangent space defined by the group elements $\bar{\mathcal{X}}_2, \bar{\mathcal{X}}_3$.

Measurement $\mathbf{z}$ may belong to a manifold. In such case, we can compute the residual as:

$$r(\bar{\mathcal{X}}, \mathbf{z}) = \mathbf{z} \diamond h(\bar{\mathcal{X}}) \,, \tag{5.46}$$

where $\diamond$ is a notation from [Solà et al., 2018], which is an operation on the composite manifold ($\ominus$ operation is applied to each of the blocks of the composite separately).

5.3.3 Belief Update at a Variable Node

At any stage, we can calculate a new marginal distribution at a variable node $\mathcal{X}_i$ by taking the product of all the incoming messages from its neighbours. If our variable belongs to a manifold

$\mathcal{M}$, similar to the factor linearisation, we need to work in the tangent space. As a common tangent plane, we create one around the previously calculated belief mean, $\bar{\mathcal{X}}_{i,t}$. All the incoming messages $m_{f_\alpha \to \mathcal{X}_i} = \mathcal{N}(\bar{\mathcal{X}}_{f_\alpha \to \mathcal{X}_i}, \mathbf{\Lambda}^{-1}_{f_\alpha \to \mathcal{X}_i})$ from the adjacent factors will be transformed into this tangent space $T_{\mathcal{X}_{i,t}}\mathcal{M}$. To transform the means of the incoming messages, we perform:

$$\bar{\boldsymbol{\tau}}_\alpha = \bar{\mathcal{X}}_{f_\alpha \to \mathcal{X}_i} \ominus \bar{\mathcal{X}}_{i,t} \in T_{\mathcal{X}_{i,t}}\mathcal{M} \,, \tag{5.47}$$

and the precision is transformed as:

$$\mathbf{\Lambda}_{\boldsymbol{\tau}_\alpha} = \mathbf{J}_r^\top(\bar{\boldsymbol{\tau}}_\alpha)\mathbf{\Lambda}_{f_\alpha \to \mathcal{X}_i}\mathbf{J}_r(\bar{\boldsymbol{\tau}}_\alpha) \,, \tag{5.48}$$

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

$$\mathbf{J}_r(\bar{\boldsymbol{\tau}}) = \frac{D\mathrm{Exp}(\bar{\boldsymbol{\tau}})}{D\bar{\boldsymbol{\tau}}} = \lim_{\epsilon \to 0} \frac{\mathrm{Exp}(\bar{\boldsymbol{\tau}} + \epsilon) \ominus \mathrm{Exp}(\bar{\boldsymbol{\tau}})}{\epsilon} \,. \tag{5.49}$$

Such transformation is derived from:

$$\begin{aligned}
\boldsymbol{\tau}_\alpha &= \mathcal{X}_\alpha \ominus \bar{\mathcal{X}}_{i,t} \,, && (5.50)\\
&= (\bar{\mathcal{X}}_\alpha \oplus \xi_\alpha) \ominus \bar{\mathcal{X}}_{i,t} \,, && (5.51)\\
&= \mathrm{Log}(\bar{\mathcal{X}}_{i,t}^{-1} \circ \bar{\mathcal{X}}_\alpha \circ \mathrm{Exp}(\xi_\alpha)) \,, && (5.52)\\
&\approx \bar{\mathcal{X}}_\alpha \ominus \bar{\mathcal{X}}_{i,t} + \mathbf{J}_r(\bar{\boldsymbol{\tau}})^{-1}\xi_\alpha \,, && (5.53)
\end{aligned}$$

using an approximation (which holds for small $\delta\bar{\boldsymbol{\tau}}$):

$$\mathrm{Log}\left(\mathrm{Exp}(\bar{\boldsymbol{\tau}})\mathrm{Exp}(\delta\bar{\boldsymbol{\tau}})\right) \approx \bar{\boldsymbol{\tau}} + \mathbf{J}_r(\boldsymbol{\tau})^{-1}\delta\bar{\boldsymbol{\tau}} \,, \tag{5.54}$$

from [Solà et al., 2018, Equation 70]. ξ_α in Equation 5.53 is a zero meaned random variable $\xi_\alpha \sim \mathcal{N}(0, \mathbf{\Sigma}_\alpha)$; hence the covariance of $\boldsymbol{\tau}_\alpha \sim \mathcal{N}(\bar{\boldsymbol{\tau}}_\alpha, \mathbf{\Sigma}^{-1}_{\boldsymbol{\tau}_\alpha})$ is:

$$\mathbf{\Sigma}_{\boldsymbol{\tau}_\alpha} = \mathbf{J}_r(\bar{\boldsymbol{\tau}})^{-1}\mathbf{\Sigma}_{f_\alpha \to \mathcal{X}_i}\mathbf{J}_r(\bar{\boldsymbol{\tau}})^{-\top} \,, \tag{5.55}$$

and taking the inverse yields us Equation 5.48.

Now we know how to transform the incoming messages and that they are all represented in a common tangent plane, we can perform a standard belief update to compute the marginal belief; however, unlike in standard GBP, here the belief is the increment:

$$\boldsymbol{\eta}_{\boldsymbol{\tau}_i} = \sum_{\alpha \in n(\mathcal{X}_i)} \mathbf{\Lambda}_{\boldsymbol{\tau}_\alpha}\bar{\boldsymbol{\tau}}_\alpha, \quad \mathbf{\Lambda}_{\boldsymbol{\tau}_i} = \sum_{\alpha \in n(\mathcal{X}_i)} \mathbf{\Lambda}_{\boldsymbol{\tau}_\alpha} \,, \tag{5.56}$$

where the computed belief of the increment is $\mathcal{N}^{-1}(\boldsymbol{\tau}_i; \boldsymbol{\eta}_{\boldsymbol{\tau}_i}, \mathbf{\Lambda}_{\boldsymbol{\tau}_i})$. To update the actual state of the variable, we retract the mean of the computed increment:

$$\bar{\mathcal{X}}_{i,t+1} = \bar{\mathcal{X}}_{i,t} \oplus \bar{\boldsymbol{\tau}}_i \,, \tag{5.57}$$

and the corresponding uncertainty gets transformed similarly to Equation 5.48:

$$\mathbf{\Lambda}_{\mathcal{X}_{i,t+1}} = \mathbf{J}_r^{-\top}(\bar{\boldsymbol{\tau}}_i)\mathbf{\Lambda}_{\boldsymbol{\tau}_i}\mathbf{J}_r^{-1}(\bar{\boldsymbol{\tau}}_i) \,, \tag{5.58}$$

and such transformation is found by letting $\boldsymbol{\tau} \sim \mathcal{N}(\bar{\boldsymbol{\tau}}_i, \boldsymbol{\Sigma}_{\bar{\boldsymbol{\tau}}_i}) = \bar{\boldsymbol{\tau}} + \mathcal{N}(0, \boldsymbol{\Sigma}_{\bar{\boldsymbol{\tau}}_i})$ and:

$$\begin{aligned}
\mathcal{X}_{i,t+1} &= \bar{\mathcal{X}}_{i,t} \oplus \boldsymbol{\tau}_i \ , && (5.59)\\
&= \bar{\mathcal{X}}_{i,t} \circ \mathrm{Exp}(\bar{\boldsymbol{\tau}}_i + \xi_i) \ , && (5.60)\\
&\approx \bar{\mathcal{X}}_{i,t} \circ \mathrm{Exp}(\bar{\boldsymbol{\tau}}_i) \circ \mathrm{Exp}(\mathbf{J}_r(\bar{\boldsymbol{\tau}}_i)\xi_i) \ , && (5.61)\\
&= \bar{\mathcal{X}}_{i,t+1} \oplus \mathbf{J}_r(\bar{\boldsymbol{\tau}}_i)\xi_i \ , && (5.62)
\end{aligned}$$

where $\xi_i \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\tau}_i})$, and we've used an approximation (which holds for small $\delta\bar{\boldsymbol{\tau}}$):

$$\mathrm{Exp}(\bar{\boldsymbol{\tau}} + \delta\bar{\boldsymbol{\tau}}) \approx \mathrm{Exp}(\bar{\boldsymbol{\tau}})\mathrm{Exp}\left(\mathbf{J}_r(\bar{\boldsymbol{\tau}})\delta\bar{\boldsymbol{\tau}}\right) \ , \tag{5.63}$$

from [Solà et al., 2018, Equation 68]. Again, the covariance is transformed by $\mathbf{J}_r(\bar{\boldsymbol{\tau}})$, and by taking the inverse, we get Equation 5.58.

To summarise, to update a variable $\mathcal{X}_i \in \mathcal{M}$:

1. Transform all incoming messages to the tangent space of the previous belief $\bar{\mathcal{X}}_{i,t}$ using Equation 5.47 and Equation 5.48.
2. Compute the incremental belief $\boldsymbol{\tau}_i \in T_{\mathcal{X}_{i,t}}$ using Equation 5.56.
3. Apply the increment to the previous belief to obtain the new belief Equation 5.57 and Equation 5.58.

5.3.4 Variable to Factor Message

A variable-to-factor message is a product of all the incoming factor-to-variable messages $m_{f_\beta \to \mathcal{X}_i} = \mathcal{N}(\bar{\mathcal{X}}_{f_\beta \to \mathcal{X}_i}, \boldsymbol{\Lambda}^{-1}_{f_\beta \to \mathcal{X}_i})$ except for the one from the outgoing factor f_α. This can be viewed as downdating the belief with the message from f_α, $m_{f_\alpha \to \mathcal{X}_i} = \mathcal{N}(\bar{\mathcal{X}}_{f_\alpha \to \mathcal{X}_i}, \boldsymbol{\Lambda}^{-1}_{f_\alpha \to \mathcal{X}_i})$.

For the Lie group extension, we follow the same idea and downdate the incremental belief $\boldsymbol{\tau}_i \sim \mathcal{N}^{-1}(\boldsymbol{\eta}_{\boldsymbol{\tau}_i}, \boldsymbol{\Lambda}_{\boldsymbol{\tau}_i})$ with the message from f_α to compute the outgoing incremental belief $m_{\boldsymbol{\tau}_\alpha \to f_\alpha} = \mathcal{N}^{-1}(\boldsymbol{\eta}_{\boldsymbol{\tau}_\alpha \to f_\alpha}, \boldsymbol{\Lambda}_{\boldsymbol{\tau}_\alpha \to f_\alpha})$:

$$\begin{aligned}
\boldsymbol{\eta}_{\boldsymbol{\tau}_\alpha \to f_\alpha} &= \boldsymbol{\eta}_{\boldsymbol{\tau}_i} - \boldsymbol{\Lambda}_{\boldsymbol{\tau}_\alpha} \left(\bar{\mathcal{X}}_{\boldsymbol{\Lambda}_{f_\alpha \to \mathcal{X}_i}} \ominus \bar{\mathcal{X}}_{i,t} \right) \ , && (5.64)\\
\boldsymbol{\Lambda}_{\boldsymbol{\tau}_\alpha \to f_\alpha} &= \boldsymbol{\Lambda}_{\boldsymbol{\tau}_i} - \boldsymbol{\Lambda}_{\boldsymbol{\tau}_\alpha} \ , && (5.65)
\end{aligned}$$

where $\boldsymbol{\Lambda}_{\boldsymbol{\tau}_\alpha} = \mathbf{J}_r^\top(\bar{\boldsymbol{\tau}}_\alpha)\boldsymbol{\Lambda}_{f_\alpha \to \mathcal{X}_i}\mathbf{J}_r(\bar{\boldsymbol{\tau}}_\alpha)$.

To compute the outgoing message, we apply the outgoing incremental belief $m_{\boldsymbol{\tau}_\alpha \to f_\alpha}$ to the current belief of the variable:

$$\begin{aligned}
\bar{\mathcal{X}}_{\mathcal{X}_i \to f_\alpha} &= \bar{\mathcal{X}}_{i,t} \oplus \left(\boldsymbol{\Lambda}^{-1}_{\mathcal{X}_i \to f_\alpha} \boldsymbol{\eta}_{\mathcal{X}_i \to f_\alpha} \right) = \bar{\mathcal{X}}_{i,t} \oplus \bar{\boldsymbol{\tau}}_{\mathcal{X}_i \to f_\alpha} \ , && (5.66)\\
\boldsymbol{\Lambda}_{\mathcal{X}_i \to f_\alpha} &= \mathbf{J}_r^{-\top}(\bar{\boldsymbol{\tau}}_{\mathcal{X}_i \to f_\alpha})\boldsymbol{\Lambda}_{\mathcal{X}_i \to f_\alpha}\mathbf{J}_r^{-1}(\bar{\boldsymbol{\tau}}_{\mathcal{X}_i \to f_\alpha}) \ , && (5.67)
\end{aligned}$$

In summary, at each iteration of GBP, variable $\mathcal{X}_i$ sends to factor f_α, the new linearisation point $\bar{\mathcal{X}}_{i,t+1}$ and the message $m_{\mathcal{X}_i \to f_\alpha} = \mathcal{N}(\bar{\mathcal{X}}_{\mathcal{X}_i \to f_\alpha}, \boldsymbol{\Lambda}^{-1}_{\mathcal{X}_i \to f_\alpha})$.

5.3.5 Factor to Variable Message

A factor-to-variable message is a marginal of a product of the factor potential from Section 5.3.2 and all the incoming variable-to-factor messages $m_{\mathcal{X}_j \to f_\alpha} = \mathcal{N}(\bar{\mathcal{X}}_{\mathcal{X}_j \to f_\alpha}, \mathbf{\Lambda}^{-1}_{\mathcal{X}_j \to f_\alpha})$ except for the one from the outgoing variable $\mathcal{X}_i$. Again, following the same ideas as variable-to-factor message derivation, we work in a common tangent plane, with increments rather than full states.

For all the incoming variable-to-factor messages, we compute the increment with respect to the current linearisation points $\bar{\mathcal{X}}_{j,t}$:

$$\bar{\boldsymbol{\tau}}_i = \bar{\mathcal{X}}_{\mathcal{X}_j \to f_\alpha} \ominus \bar{\mathcal{X}}_{j,t} \,, \tag{5.68}$$

$$\mathbf{\Lambda}_{\boldsymbol{\tau}_i} = \mathbf{J}_r^\top(\bar{\boldsymbol{\tau}}_i) \mathbf{\Lambda}_{\mathcal{X}_j \to f_\alpha} \mathbf{J}_r(\bar{\boldsymbol{\tau}}_i) \,, \tag{5.69}$$

applying the transformations to the precision as we did in Equation 5.58.

Now, we need to take the product of all the messages but one and the factor potential and then marginalise. To illustrate, consider a case where we have a factor connected to one variable in $\mathbb{R}^2$ and two variables in $\mathcal{M}$. After applying linearisation (Section 5.3.2), the information vector and the precision matrix will be:

$$\boldsymbol{\eta}_f = \begin{pmatrix} \boldsymbol{\eta}_{f_1} \\ \boldsymbol{\eta}_{f_2} \\ \boldsymbol{\eta}_{f_3} \end{pmatrix} = \mathbf{J}^\top \mathbf{\Lambda} (\mathbf{z} \diamond h(\bar{\mathcal{X}}_t)), \tag{5.70}$$

$$\mathbf{\Lambda}_f = \begin{bmatrix} \mathbf{\Lambda}_{f_{11}} & \mathbf{\Lambda}_{f_{12}} & \mathbf{\Lambda}_{f_{13}} \\ \mathbf{\Lambda}_{f_{21}} & \mathbf{\Lambda}_{f_{22}} & \mathbf{\Lambda}_{f_{23}} \\ \mathbf{\Lambda}_{f_{31}} & \mathbf{\Lambda}_{f_{32}} & \mathbf{\Lambda}_{f_{33}} \end{bmatrix} = \mathbf{J}^\top \mathbf{\Lambda} \mathbf{J} \tag{5.71}$$

If we choose the output variable to be the third variable, we need to first combine with the factor potential the incoming messages from variables $\mathcal{X}_1$ and $\mathcal{X}_2$:

$$\boldsymbol{\eta}'_f = \begin{pmatrix} \boldsymbol{\eta}_{f_1} + \mathbf{\Lambda}_{\boldsymbol{\tau}_1} \bar{\boldsymbol{\tau}}_1 \\ \boldsymbol{\eta}_{f_2} + \mathbf{\Lambda}_{\boldsymbol{\tau}_2} \bar{\boldsymbol{\tau}}_2 \\ \boldsymbol{\eta}_{f_3} \end{pmatrix} \tag{5.72}$$

$$\mathbf{\Lambda}'_f = \begin{bmatrix} \mathbf{\Lambda}_{f_{11}} + \mathbf{\Lambda}_{\boldsymbol{\tau}_1} & \mathbf{\Lambda}_{f_{12}} & \mathbf{\Lambda}_{f_{13}} \\ \mathbf{\Lambda}_{f_{21}} & \mathbf{\Lambda}_{f_{22}} + \mathbf{\Lambda}_{\boldsymbol{\tau}_2} & \mathbf{\Lambda}_{f_{23}} \\ \mathbf{\Lambda}_{f_{31}} & \mathbf{\Lambda}_{f_{32}} & \mathbf{\Lambda}_{f_{33}} \end{bmatrix} \tag{5.73}$$

To complete message passing, from this joint distribution, we must marginalise out $\mathcal{X}_1$ and $\mathcal{X}_2$ to obtain an incremental marginal belief of the outgoing variable $\mathcal{X}_3$. Using Equation 2.12:

$$\boldsymbol{\eta}_{f_\alpha \to \boldsymbol{\tau}_3} = \boldsymbol{\eta}_{f_3} - \begin{pmatrix} \mathbf{\Lambda}_{f_{31}} & \mathbf{\Lambda}_{f_{32}} \end{pmatrix} \begin{bmatrix} \mathbf{\Lambda}_{f_{11}} + \mathbf{\Lambda}_{\boldsymbol{\tau}_1} & \mathbf{\Lambda}_{f_{12}} \\ \mathbf{\Lambda}_{f_{21}} & \mathbf{\Lambda}_{f_{22}} + \mathbf{\Lambda}_{\boldsymbol{\tau}_2} \end{bmatrix}^{-1} \begin{pmatrix} \boldsymbol{\eta}_{f_1} + \mathbf{\Lambda}_{\boldsymbol{\tau}_1} \bar{\boldsymbol{\tau}}_1 \\ \boldsymbol{\eta}_{f_2} + \mathbf{\Lambda}_{\boldsymbol{\tau}_2} \bar{\boldsymbol{\tau}}_2 \end{pmatrix} , \tag{5.74}$$

$$\mathbf{\Lambda}_{f_\alpha \to \boldsymbol{\tau}_3} = \begin{pmatrix} \mathbf{\Lambda}_{f_{31}} & \mathbf{\Lambda}_{f_{32}} \end{pmatrix} \begin{bmatrix} \mathbf{\Lambda}_{f_{11}} + \mathbf{\Lambda}_{\boldsymbol{\tau}_1} & \mathbf{\Lambda}_{f_{12}} \\ \mathbf{\Lambda}_{f_{21}} & \mathbf{\Lambda}_{f_{22}} + \mathbf{\Lambda}_{\boldsymbol{\tau}_2} \end{bmatrix}^{-1} \begin{pmatrix} \mathbf{\Lambda}_{f_{12}} \\ \mathbf{\Lambda}_{f_{23}} \end{pmatrix} , \tag{5.75}$$

and, before sending the message, we convert the message into a full state using retraction:

$$\bar{\mathcal{X}}_{f_\alpha \to \mathcal{X}_3} = \bar{\mathcal{X}}_{3,t} \oplus \mathbf{\Lambda}^{-1}_{f_\alpha \to \boldsymbol{\tau}_3} \boldsymbol{\eta}_{f_\alpha \to \boldsymbol{\tau}_3} = \bar{\mathcal{X}}_{3,t} \oplus \bar{\boldsymbol{\tau}}_{f_\alpha \to \boldsymbol{\tau}_3} \,, \tag{5.76}$$

$$\mathbf{\Lambda}_{f_\alpha \to \mathcal{X}_3} = \mathbf{J}_r^{-\top}(\bar{\boldsymbol{\tau}}_{f_\alpha \to \boldsymbol{\tau}_3}) \mathbf{\Lambda}_{\bar{\boldsymbol{\tau}}_{f_\alpha \to \boldsymbol{\tau}_3}} \mathbf{J}_r^{-1}(\bar{\boldsymbol{\tau}}_{f_\alpha \to \boldsymbol{\tau}_3}) \,, \tag{5.77}$$

5.4 Summary

In this section, we followed the derivations from [Davison and Ortiz, 2019] to extend BP to GBP, and demonstrated how GBP can be modified to handle non-linearity. We then extend their work further by adding robust M-Estimators following iteratively reweighted least squares and applying Lie theory to handle Lie group variables. Such extensions are critical for many robotics applications since real-world observations contain outliers and state space of robots are often modelled as $\mathbf{SE}(2)$ or $\mathbf{SE}(3)$.

CHAPTER 6

A Robot Web for Distributed Many-Device Localisation

Contents

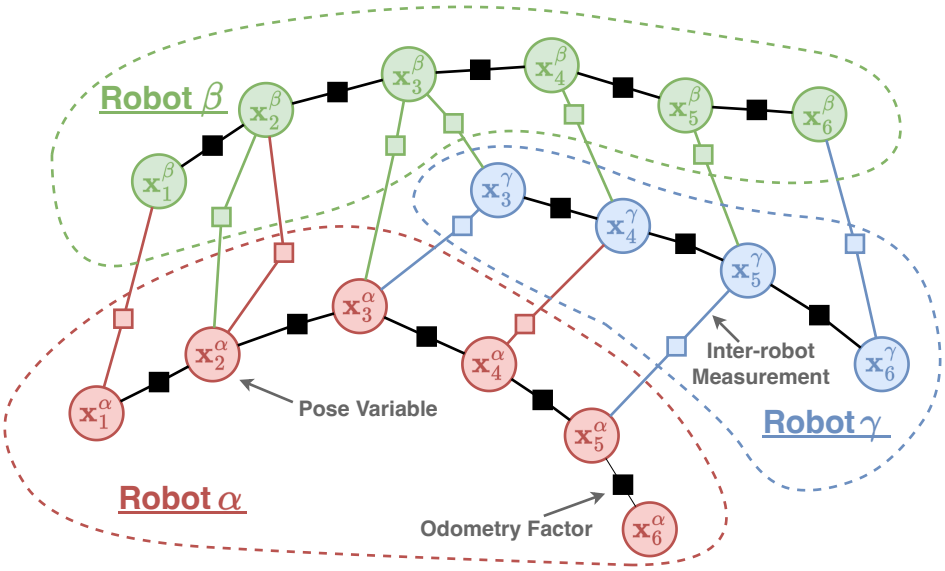


Figure 6.1: In the Robot Web, we assume that a set of robots move through space while using their sensors to observe each other. The circles represent the variables – where $\mathbf{x}_t^\alpha$ denotes a variable at timestamp t which belongs to robot α –, and the squares are the factors. Robot γ starts at timestamp 3 for clarity of visualisation. The full-factor graph for multi-robot localisation is used. Responsibility for storing and updating it is divided up between the multiple robots participating, as shown by the coloured regions separated by dotted lines. Each robot maintains its own pose variable nodes, odometry factors, and factors for the inter-robot measurements made by its sensors, and carries out continuous GBP on this graph fragment. Message passing across dotted line boundaries happens on an asynchronous and ad-hoc basis.

We show that a distributed network of robots or other devices which make measurements of each other can collaborate to globally localise via efficient ad-hoc peer-to-peer communication. Our Robot Web solution is based on Gaussian Belief Propagation on the fundamental non-linear factor graph describing the probabilistic structure of all of the observations robots make internally or of each other, and is flexible for any type of robot, motion or sensor. We define a simple and efficient communication protocol which can be implemented by the publishing and reading of web pages or other asynchronous communication technologies. We show in simulations with up to 1000 robots interacting in arbitrary patterns that our solution convergently achieves global accuracy as accurate as a centralised non-linear factor graph solver while operating with high distributed efficiency of computation and communication. Via the use of robust factors in GBP, our method is tolerant to a high percentage of faulty sensor measurements or dropped communication packets. Furthermore, we showcase that the system operates on real robots with limited onboard computational resources.

6.1 Introduction

As we head towards a future where embodied artificial intelligence is ubiquitous, we expect that multiple robots, vehicles and other devices which share the same environment will need to communicate and coordinate their actions, whether their goal is explicit cooperation or just safe independent action. One clear possibility is that all devices could use a unified cloud-based 'maps' system, presumably owned by one company or government, which tracks and coordinates all devices. An alternative, which we investigate here, is a distributed system-based on per-device local computation and storage, and peer-to-peer communication between heterogeneous devices from different makers using standardised open protocols. Inspired by the original design of the World Wide Web, we call this concept the **Robot Web**.

A key outstanding problem in multi-robot systems has been true distributed localisation: how can a set of moving devices which move and observe each other within a space estimate their locations, using noisy actuators, sensors and realistic peer-to-peer communication?

In this chapter, we present Robot Web, a solution to general, fully distributed and asynchronous many-robot localisation. Our solution is based on the fundamental probabilistic factor graph representation of perception and state estimation. We show that Gaussian Belief Propagation (GBP) [Ortiz et al., 2021] is the key algorithm with the appropriate properties of distributed processing, storage and message passing, which permits inference on a dynamically changing estimation problem via ad-hoc communication between robot peers.

In our solution, each robot stores and maintains its own part of the full factor graph (as shown in Figure 6.1), and updates and publishes a Robot Web Page of outgoing messages for other robots to download and read whenever possible. Remarkably, using GBP the whole factor graph can efficiently converge to localisation estimates as accurate as full batch optimisation but without any device ever needing to store or process more than its own local graph fragment. Robots communicate via ad-hoc, asynchronous messages containing only small vectors and matrices. Significantly, GBP can deal with graphs which have any type of parameterisation (*e.g.* 2D or 3D robot movement, or any type of non-linear sensor measurements) and which can change dynamically in arbitrary ways — for instance, robots can join or leave the web whenever needed, or reconfigure their sensors online. We will show that it can also cope with and reject a high fraction of outlier measurements, for instance, caused by faulty sensors, and deal with highly unreliable communication channels.

Our approach is designed for scalability. All communication is via a simple interface, and robots do not need any privileged information about each other, such as even how many other robots are involved. The whole Robot Web therefore can be fully dynamic, with robots joining or leaving at will. We believe that this formulation of many-robot localisation could be the foundation for a new era of distributed Spatial AI.

To summarise, the key contributions of our work are:

- A distributed, scalable localisation system based on Gaussian Belief Propagation (GBP) — **Robot Web** — that can localise thousands of robots in a simulated environment. Our method naturally handles asynchronous communication, communication failures, and a large number of outlier sensor measurements. Additionally, because all processing is local, our method can adapt to dynamic changes in the topology of the graph and handle disjoint connectivity.

- Extensive evaluation of our system to demonstrate the scalability. Using simulation, comparisons are made against a centralised counterpart across a range of parameters. Additionally, we have tested our system under challenging conditions, including the presence of a large number of non-Gaussian outlier sensor measurements and communication failures, in order to demonstrate its robustness.

- Real-time experiments with nine physical robots using only the limited computational resources available onboard. These experiments demonstrate the practicality and effectiveness of our approach and demonstrate that our method is feasible in real-world applications.

6.2 Related Work

Robot Web uses the standard Gaussian factor graph representing the multi-robot localisation problem and is most closely related to the wealth of factor graph formulations and solvers in robotics, as well explained in the work of Dellaert and Kaess [Dellaert and Kaess, 2017, Dellaert, 2021]. Most methods for inference on factor graphs assume a centralised computer with access to the whole graph and focus on either efficient batch solution or incremental inference on graphs that are continually changing. Centralised pose-graph optimisation algorithms suitable for multiple robots are well-explored in the literature [Indelman et al., 2014, Bailey et al., 2011, Kim et al., 2010, Andersson and Nygards, 2008]. MR-iSAM2 [Zhang et al., 2021] extends iSAM2 [Kaess et al., 2012] to build an incremental, centralised graph optimisation method for multiple robots. However, centralised methods require a base station, and are vulnerable to failure of this station, can require high communication bandwidth, create privacy concerns, and generally are not scalable [Lajoie et al., 2022]. Many distributed, multi-robot localisation where attempted using many distributed algorithms. Such algorithms and literature are summarised in Section 1.5.

Many recent advancements in multi-robot localisation leverage the advancements in distributed pose-graph optimisation (PGO). For example, [Choudhary et al., 2017] uses chordal-relaxation to make the underlying PGO problem linear, and solves them using a Gauss-Seidel solver. This work is used as a backend optimiser in many distributed SLAM systems [Lajoie et al., 2020, Cieslewski et al., 2018]. Semidefinite Programming (SDP) relaxation together with Riemannian block coordinate descent is used in [Tian et al., 2020, Tian et al., 2021] which enables verification of the correctness of the estimates, and is decentralised and asynchronous. A distributed SLAM system [Tian et al., 2022] is built upon these approaches, demonstrating their practicality. However, the above approaches require a full relative transformation between the robots and can only handle

isotropic covariance. This could be limiting, for example, if the inter-robot observations are all range-bearing, as we will explore in this chapter. More recently, methods use range measurements [Papalia et al., 2023] or bearing measurements [Wang et al., 2022b] and show that it is possible to obtain certifiably optimal solutions again via SDP relaxation. While all PGO-based methods achieve good localisation accuracy, the formulation is often tailored and these methods do not generalise to other problem instances.

More general methods for multi-robot localisation such as DDF-SAM [Cunningham et al., 2010], and DDF-SAM2 [Cunningham et al., 2013] operate on factor graphs. They rely on Gaussian elimination and require robots to exchange Gaussian marginals about shared variables. The communication; however, increases quadratically with the number of shared variables. Alternating Direction Method of Multipliers (ADMM) has been employed for distributed SLAM [McGann et al., 2023] or to efficiently share map points for distributed bundle adjustment [Bänninger et al., 2023]. However, unlike elimination-based approaches only the point-estimates are recovered.

A significant limitation of the distributed methods above is that they are synchronous, which means the robots must share their messages at predetermined times to ensure the shared information is up-to-date. In contrast, asynchronous methods offer the flexibility that the robots can operate at their own rate, without waiting for other robots. Examples includes, [Todescato et al., 2015] and [Tian et al., 2020], and the paper discusses how the algorithms behave under bounded message delays.

Robot Web is also asynchronous and distributed, but both much more simple in formulation and more general than these methods. Significantly, [Tian et al., 2020] assume Gaussian noise is unable to handle outliers. Additionally, their formulation requires the sensor measurements to be a relative transformation. Robot Web supports general robot and sensor models and allows robust factors, making it robust to large fractions of non-Gaussian outlier measurements.

There has been some work on multi-agent distributed localisation using variations of belief propagation in the sensor networks community. For instance, [Schiff et al., 2009] performed multi-robot localisation using non-parametric belief propagation. In [Wymeersch et al., 2009] belief propagation was also used to perform cooperative positioning in a distributed manner, performing a sum-product algorithm over a factor graph in an ultra-wideband network. In [Caceres et al., 2011], [Wymeersch et al., 2009] was extended to a network composed of GNSS nodes. In [Li et al., 2014] and [Li et al., 2015], non-linear range measurements were solved by linear approximation. In [Wan et al., 2017] a hybrid parametric and non-parameteric Belief Propagation (BP) was proposed to model the non-linearity and any non-Gaussian distributions. However, the method requires sampling and does not exhibit a close-form expression.

Robot Web goes far beyond these methods to present a general framework for general robots and sensors. It defines for the first time an open, asynchronous communication framework, and via the focus on GBP with robust factors enables highly robust and scalable performance.

6.3 Gaussian Belief Propagation

Robot Web's core algorithm is GBP, which we've derived in Chapter 5. GBP performs marginal inference iteratively via message passing between variable and factor nodes, which can happen in many different message-passing schedules but still with convergent behaviour. In this section, we summarise how we can use GBP across many devices for the task of distributed localisation.

6.3.1 Gaussian Belief Propagation with Lie Groups

It is possible to use Robot Web in vector space with no modification to the GBP as introduced in Section 5.1. However, in most realistic robotics problems, there are additional details to consider due to the state space being a robot pose with rotation and translation, where careful thought about parameterisation is needed. The use of Lie theory is a key component of modern state estimation for robotics [Barfoot, 2017], and is applied to algorithms such as Extended Kalman Filter, and Information Filter [Ćesić et al., 2017], and thus, as derived in Section 5.3, we use an extension of GBP, where the variables can be Lie group elements. This allows us to handle poses properly without worrying about singularities.

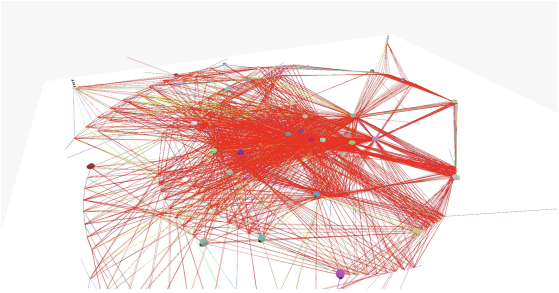
6.3.2 Distributed Gaussian Belief Propagation

Extending GBP to the distributed multi-device system is straightforward. Since GBP is a node-wise distributed algorithm and it operates via message passing, no change to the core algorithm is required for distributed inference.

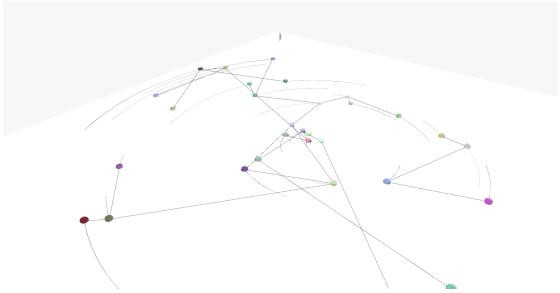
Let G be the global factor graph which we want to perform inference over. In distributed GBP, robots own a factor graph G^ω each, and their union is $G = \bigcup_{\omega\in\Omega} G^\omega$. Each robot *has ownership* of its pose variables and the factors corresponding to the observations it made, i.e. the nodes that their local graph G^ω consists of. This is important as the global factor graph G is partitioned amongst the robots Ω; hence, the marginal estimates obtained by solving the distributed problem using GBP are *exactly* the same as the marginal estimates of the global problem obtained via centralised GBP under an assumption of perfect communication.

Distributed inference is achieved by each robot $\alpha \in \Omega$ performing GBP message passing on their local graph G^α. Along the edges of a factor $g^{\alpha,\beta}$ owned by α, a factor-to-variable message $m_{g^{\alpha,\beta}\to x^\beta}(x^\beta)$ (Section 5.1.4) is sent from α to β via inter-robot communication. Similarly, β sends back to α variable-to-factor message, $m_{x^\beta\to g^{\alpha,\beta}}(x^\beta)$ (Section 5.1.3).

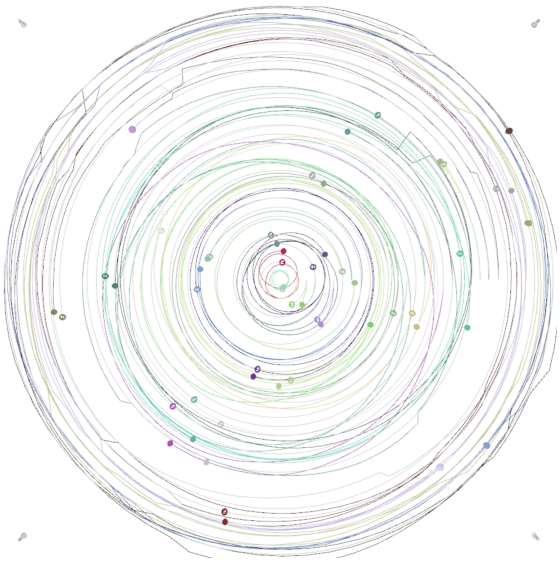
Such a concept is depicted in Figure 6.1, where the graph is partitioned amongst the robots. Using Robot α and Robot β from the figure as an example, the factors of Robot α will send factor-to-variable messages to $\mathbf{x}_1^\beta$, $\mathbf{x}_2^\beta$, and variable-to-factor messages from $\mathbf{x}_2^\alpha$, $\mathbf{x}_3^\alpha$ together with its point-estimate $\mu_{\mathbf{x}_2^\alpha}$, $\mu_{\mathbf{x}_3^\alpha}$ as the linearisation point. Both factor-to-variable and variable-to-factor



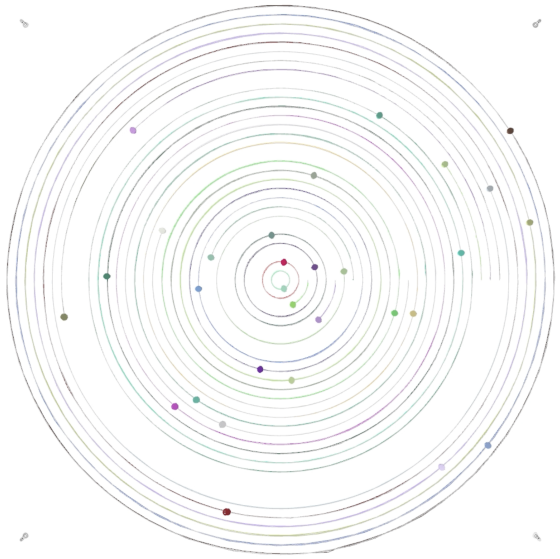
(a) Observations between devices.



(b) Communication pattern between devices.



(c) Localisation using only four landmarks.



(d) Localisation with inter-robot observations.

Figure 6.2: Visualisation of Robot Web. Robots are instructed to follow a circular path. Four landmarks are positioned at the corner. As the odometry of the robots is noisy, they cannot stay on a circular path without localising and correcting for the drift.**Top Row:** We visualise the inter-device observations with red lines (Left), and inter-device communication with black lines (Right). **Bottom Row:** (Left:) robots only use the landmarks for localisation. (Right:) robots use inter-robot observations to localise.

messages are a marginal distribution, which is a $N \times 1$ vector and a $N \times N$ precision matrix, where N is the Degrees of Freedom (DoF) of the variable.

6.4 Robot Web: Core Design and Structure

We will use the term 'robot' for any device involved in the Robot Web, but some of these could be beacons, sensor nodes, or any other type of participating entity, which could be moving or stationary.

In Figure 6.2, many devices perform localisation to maintain a circular path. As demonstrated, robots fail to maintain the path when relying solely on landmark-based localisation, yet they successfully maintain the path when utilising inter-robot observations for localisation. This highlights the core goal of Robot Web, which is to enhance devices' spatial AI capabilities beyond each

device's perceptual capabilities.

6.4.1 Partitioning of the Factor Graph

The fundamental structure of the Robot Web is the full probabilistic factor graph which represents the states of robots as variables and the measurements they make, or any other information which is available such as pose or smoothness priors, as factors. Determining estimates of the robot states is a matter of performing inference on this factor graph to produce marginal distributions over the variables. We will assume that all factors take the form of Gaussian functions of the involved state variables, and use Gaussian Belief Propagation as the mechanism for inference. Note that GBP supports robust (heavy-tailed) factors and non-linear measurement functions via the methods proposed in [Davison and Ortiz, 2019], and therefore this model is very broadly practically applicable. These are the same assumptions behind most centralised factor graph inference libraries, such as GTSAM [Dellaert, 2012], Ceres [Agarwal et al., 2010], and g2o [Kümmerle et al., 2011].

The key concept of the Robot Web is to distribute responsibility for storing and updating the full-factor graph, by dividing it up between the robots taking part. Figure 6.1 illustrates this for an elementary case of three moving robots, each with internal odometry sensing and an outward-looking sensor able to make observations of the other robots. We use different colours to highlight the parts of the factor graph for which each robot is responsible. A Robot α stores:

- The set of variables $\mathbf{x}_t^\alpha$ representing its state at discrete times t. It could store a whole history of states or a finite window. Most commonly these states will be multi-dimensional variables which directly represent robot pose, though any other aspects of internal state could be included. Pose is parameterised using $\mathbf{SE}(2)$ as discussed in Section 5.3.

- The set of factors f_t^α or $f_{t,t+1}^\alpha$ representing priors or internal measurements. Each of these factors connects to one or more of the robot's own state variables. Common examples would be a unary factor representing a GPS pose measurement, or a binary factor connecting two temporally consecutive states representing an odometry or inertial measurement.

- A set of factors $g_t^{\alpha,\beta}$ representing observations made by robot α of other β. Specifically, it represents a measurement made by this robot α of another robot β at time t. This factor connects one state variable $\mathbf{x}_t^\alpha$ from robot α with state variable $\mathbf{x}_t^\beta$ of robot β at the corresponding time.

There is an important design choice here: factors representing inter-robot measurements are stored by the robot making the measurement. This is because the details of measurement factors depend on the type and calibration of the sensor involved, and in this way, those details only need to be known to the robot carrying the sensor. Note also that we assume for now that all robots have globally synchronised clocks for timestamping of measurements (though we will see that all computation and communication can be asynchronous).

The factor graph evolves and grows dynamically. At initialisation, a robot will have just one variable node. As it moves, and measures its own incremental motion with odometry or similar, it adds the appropriate variables and factors to its internal factor graph. GBP runs continuously on the robot's internal factor graph, producing always-updating marginal distributions for each variable. The message passing pattern of GBP within a robot's internal factor graph is not important but should be rapid and global enough to keep the graph fragment mostly close to convergence.

6.4.2 Message Passing and Communication Model

When the robot uses an outward-looking sensor to make an observation of the relative location of another robot, it creates a factor for this measurement, connects it to its current live pose variable, and the factor takes part in local GBP. The other end of this factor will initially be unconnected, because the appropriate variable to attach it to is stored by another robot: the factor-to-variable edge crosses the 'dotted line' boundary (see Figure 6.1), separating factor graph fragments. When local GBP generates an outgoing message from this factor which crosses the dotted line, that message is made available to the other robot that needs it.

The key idea behind Robot Web is that its inter-robot communication model is flexible and does not require synchronous or bidirectional communication. Instead, each robot can broadcast information at its own rate, which is particularly useful in large-scale systems where synchronising communication across multiple robots can be a challenge. This is only possible as GBP can converge even with an arbitrary message schedule [Ortiz et al., 2021] meaning that the communication between robots can be completely asynchronous and ad-hoc, but the overall graph made up of many fragments will converge to the global estimates. We assume that communication is the most resource-intensive operation in the overall system. The computation performed by GBP is lightweight, especially with the sliding window optimisation, and can run on devices such as Raspberry Pi without any additional optimisation. Therefore, a flexible inter-device communication protocol is important, as it allows the communication rate to be varied based on factors such as the available onboard battery.

Communication Model

The asynchronous nature of the communication allows for a variety of options for message delivery, such as the publish-subscribe model used in systems like ROS/ROS2, where devices broadcast messages and listen to topics of interest, or the pull model, where devices query each other for information. In our work, we do not assume that messages will always be delivered, and any loss of messages will only result in a possible decrease in localisation accuracy, rather than causing a deadlock or critical failure. Additionally, we stress that our approach does not involve any shared global information apart from common timestamps and unique identifiers among the robots. Each robot only exchanges messages with the others and does not share sensor models, initialisation status, or even the number of robots participating in the optimisation process. This combination of

asynchronous communication and lack of shared global information allows the system to function even if there are fewer communication rounds than the total number of robots.

Inter-device Factor Discovery

In Robot Web, the connection over the inter-device factors is formed lazily. A factor is created when an observation occurs. However, it takes a few iterations of message passing before the factor can be linearised. Specifically, when Robot α observes Robot β at timestamp t, a factor $g_t^{\alpha,\beta}$ is formed. As the observation is made by Robot α, it owns the factor. Robot α publishes an empty message $m_{g_t^{\alpha,\beta} \to \mathbf{x}_t^\beta}$. Upon Robot β receiving the message, it will publish the message $m_{\mathbf{x}_t^\beta \to g_t^{\alpha,\beta}}$ together with the linearisation point $\bar{\mathcal{X}}_t^\beta$. When Robot α receives this message, it linearises the factor and starts the optimisation process. In the succeeding rounds, Robot β will receive a message from Robot α which it can use to refine its pose estimate.

As GBP performs operations locally and does not use global information (*e.g.* topology of the graph), this sequence of message exchange occurs asynchronously, and GBP continues optimising as it discovers new inter-device factor connections.

Robot Web Page Interface

One of the main motivations for the design choices made in Robot Web is the desire for **distributed scalability**. By providing a uniform interface, the robots can be added to or removed from the Robot Web in a fully dynamic manner, or can freely change their internal methods or software as long as they maintain the same interface. The internal complexity of each robot's processing may be slightly increased because of this, but this is a small price to pay for global scalability.

To achieve this goal, one potential approach is to use a simple Web protocol (*e.g.* HTTP) for all inter-robot communication, with each robot hosting outgoing messages as a Web page. This allows inter-robot communication to happen in arbitrary patterns and in a read-only style, which can contribute to the scalability and flexibility of the system.

6.5 Demonstrations and Experiments in a Simulated Environment

We present extensive simulation demonstrations of Robot Web localisation for the case of many robots with planar 2D motion and noisy odometry and inter-robot range-bearing measurements. Our simulation uses metric units and models an application like a warehouse setting where tens or hundreds of robots roam through an environment 100m across with randomly generated paths. Usually, we add a handful of known beacon landmarks to the environment, whose positions are known in advance to all robots, but are widely spread so that robot-landmark measurements are much less frequent than robot-robot measurements. The main role of the landmarks is to anchor the whole web to an absolute coordinate frame over long periods of operation.

Our simulation uses a fully distributed program structure equivalent to what could be achieved on a true multi-robot system.

6.5.1 Implementation Details

In our experiments, we run the robots in a square arena of width 100m with 10 known beacons where all robots move through 100 pose steps. All variable nodes in the current simulation are represented using $\mathbf{SE}(2)$, and three different factors are implemented:

Anchor Factor: If needed, we can use unary anchor factors which are priors on the poses of robots before they start moving. These are only added to the initial pose of each robot to establish a rough common coordinate frame and are not added to any subsequent poses. In most experiments, we use these factors to represent fairly well-known initial robot positions at the start of motion, though note that in Section 6.5.7, we show that new robots can be added to an existing web without any pose priors. The uncertainty assigned to the anchor factors in our main experiments is: $\sigma_x = 0.1$m, $\sigma_y = 0.1$m, and $\sigma_\theta = 0.01$ rad.

Odometry Factor: The odometry factor relating the two consecutive robot poses $\mathbf{x}^\alpha_{t-1}$ and $\mathbf{x}^\alpha_t$, and the likelihood is define as:

$$l_o(\mathbf{x}^\alpha_{t-1}, \mathbf{x}^\alpha_t; \bar{\mathbf{z}}^\alpha_{t-1,t}) \propto \exp(-\frac{1}{2}\|\bar{\mathbf{z}}^\alpha \ominus h_o(\mathbf{x}^\alpha_{t-1}, \mathbf{x}^\alpha_t)\|^2_{\Sigma_o}) \,, \tag{6.1}$$

where the measurement $\bar{\mathbf{z}}^\alpha \in \mathbf{SE}(2)$. For all the simulations, we set the following uncertainty per meter step is $\sigma_x = 0.1$m, $\sigma_y = 0.01$m, and $\sigma_\theta = 0.01$ rad. For the odometry factor, we define the measurement prediction function as $h_o(\mathbf{x}^\alpha_{t-1}, \mathbf{x}^\alpha_t) = (\mathbf{x}^\alpha_{t+1})^{-1} \circ \mathbf{x}^\alpha_t$.

Range-Bearing Factor: We use a range-bearing sensor for the measurements between robots, or between robots and landmarks. We parameterise the observation $\bar{\mathbf{z}}^{\alpha,\beta}_t \in \langle \mathbb{R}, \mathbf{SO}(2) \rangle$ using polar coordinate (r, θ), i.e. radial distance and azimuthal angle, respectively.

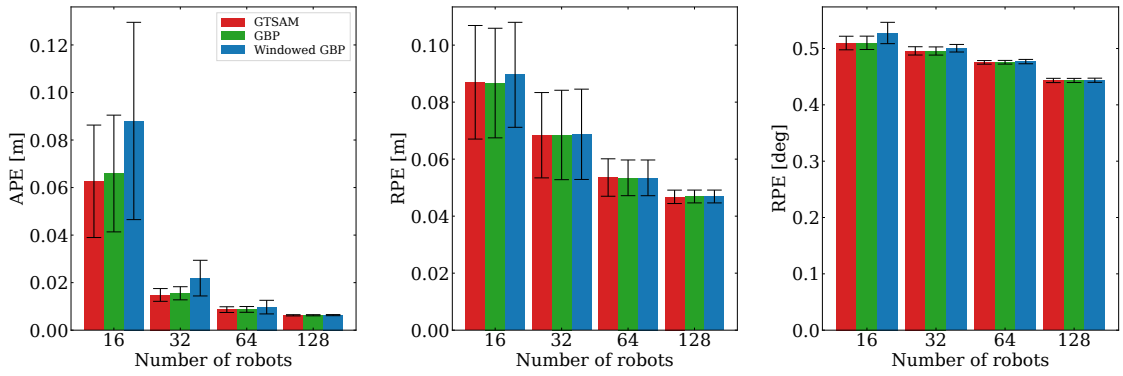
$$l_s(\mathbf{x}^\alpha_t, \mathbf{x}^\beta_t; \bar{\mathbf{z}}^{\alpha,\beta}_t) \propto \exp(-\frac{1}{2}\|\bar{\mathbf{z}}^{\alpha,\beta}_t \diamond h_s(\mathbf{x}^\alpha_t, \mathbf{x}^\beta_t)\|^2_{\Sigma_s}) \,. \tag{6.2}$$

Let $\mathbf{t}^\alpha_t \in \mathbb{R}^2$, $\mathbf{R}^\alpha_t \in \mathbf{SO}(2)$ be translational, rotational the part of $\mathbf{x}^\alpha_t$, and similarly for $\mathbf{x}^\beta_t$. The relative transformation between $\mathbf{t}^\alpha_t$ and $\mathbf{t}^\beta_t$ in the coordinate frame of α is: $\mathbf{R}^{\alpha\top}_t(\mathbf{t}^\beta_t - \mathbf{t}^\alpha_t) = (\delta x, \delta y)$. The measurement prediction function is defined as $h_s(\mathbf{x}^\alpha_t, \mathbf{x}^\beta_t) = (r, \theta)$, where $r = \|\mathbf{t}^\beta_t - \mathbf{t}^\alpha_t\|_2$ and $\theta = \text{arctan2}(\delta y, \delta x)$.

By default, the uncertainty assigned to range/bearing is: $\sigma_r = 0.01$m, $\sigma_b = 0.05$ rad, with the sensor range limited to 30m. DSC [Agarwal et al., 2012]:

$$s_m = \min\left(1, \frac{2\Phi}{\Phi + E_m}\right), \tag{6.3}$$

is used as the robust kernel with $\Phi = 10$. The factor therefore is down-weighted accordingly as described in Section 5.2.2. Alongside the Gaussian noise, to 10% of all range-bearing measurements, we additionally add a huge amount of uniform noise: $r_n \sim \mathcal{U}(0, 30)$, $b_n \sim \mathcal{U}(0, \pi)$ to simulate non-Gaussian noise which makes these measurements essentially useless.



(a) Mean RMSE ATE (b) Mean RMSE trans. RPE (c) Mean RMSE rot. RPE

Figure 6.3: In a simulated environment, N robots are moving around in an environment with 10 known landmarks for 100 poses each. GTSAM optimises the factor graph after every pose insertion rather than solving after all poses are inserted to keep the comparison fair. GBP uses the full factor graph to optimise, while Windowed GBP only uses only the last 5 poses. The results are the average of 10 runs with different random initialisation, and the error bar represents one standard deviation of uncertainty.

We use a communication pattern which simulates a limited peer-to-peer communication budget, where each robot connects to and reads the Robot Web page from other robots in a sequential, random pattern with closer robots are more likely to be selected. The idea is that this is similar to a robot sequentially switching its Wi-Fi connection between peers with strong signals.

We generated a noisy distance sample between Robot α and Robot β as $d_{\alpha,\beta} \sim \mathcal{N}(\bar{d}_{\alpha,\beta}, 0.1)$, where $d_{\alpha,\beta}$ is the random sample and $\bar{d}_{\alpha,\beta}$ is the ground truth distance between Robot α and Robot β. We define the neighbourhood of α, $N(\alpha)$, as the set of robots which Robot α can communicate with. The probability $C_{\alpha,\beta}$ that Robot α communicates with Robot $\beta \in N(\alpha)$ is:

$$p(C_{\alpha,\beta}) = \frac{1/d_{\alpha,\beta}^2}{\sum_{\omega \in N(\alpha)} 1/d_{\alpha,\omega}^2} \,. \tag{6.4}$$

In this work, we assume that $N(\cdot)$ includes all robots. Each robot performs 20 iterations of GBP per movement step and at each GBP step robots communicate with only one neighbour. Factors are dampened [Murphy et al., 1999] by 0.2, and the factors linearise at every iteration. Any changes to the default parameters will be specified in the individual experiments.

6.5.2 Convergence and Computational Properties

A key property of our method is that the marginal estimates generated by message passing with a fixed computation and communication budget on our ever-changing factor graph may not necessarily be at complete convergence during live operation, though that is often not a problem if useful robot pose estimates are still achieved. Nevertheless, here we show that when enough computation and communication are regularly applied, the localisation results are convergent and estimates as accurate as a batch solution on a centralised processor can be achieved. Importantly,

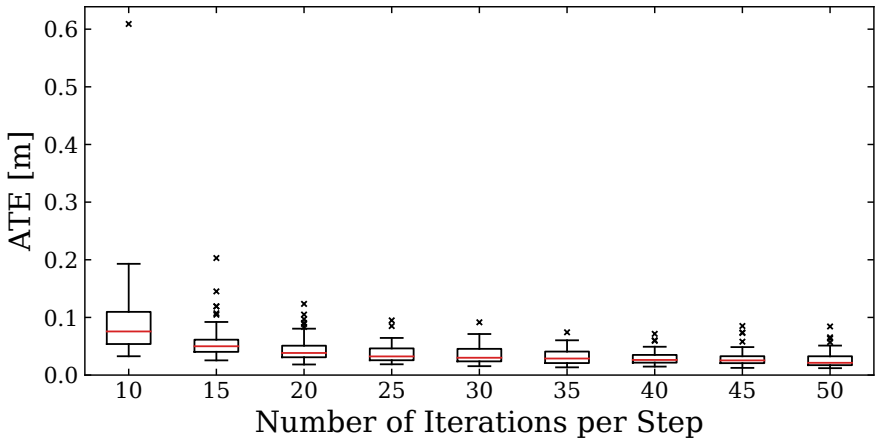


Figure 6.4: Increasing the number of iterations per step decreases the overall error. Even with a small number of iterations, GBP is able to provide good localisation, which can be further refined by increasing the iterations. The red line shows the median, the box extends from the first quartile to the third quartile, the whisker extends from the box by 1.5 inter-quartile range, and the outliers are marked with a cross. In a simulated environment, 50 robots are moving in an environment with 10 known landmarks for 100 poses each. Each result is a summary of 50 runs with different random initialisation.

this can be achieved with highly efficient, realistic settings for distributed GBP computation and communication.

Here we present an experiment to compare the accuracy of distributed Robot Web GBP against a centralised solution of the same factor graph using GTSAM [Dellaert, 2021]. GBP linearises at every 5 iterations and is allowed to optimise for 20 iterations per step. To keep the comparison simple, robust kernels are not applied for both GBP and GTSAM, and we do not add uniform noise to the range-bearing measurements.

We present results for general GBP, where each robot keeps a full history of pose variables, and Windowed GBP, where each robot maintains a sliding window of its most recent 5 poses and only processes messages relating to these. Using a sliding window allows the average size of the factor graph, and the amount of computation needed, to remain fixed. This allows the system to operate over an arbitrarily long period while maintaining constant computational cost. What we lose by doing this is the possibility to improve estimates of older variables in the graph using new observations.

In these experiments, for both versions of GBP, all robots are allowed to communicate with each other on every iteration.

We report the Root Mean Square Absolute Trajectory Error (RMSE ATE), and Root Mean Square Relative Pose Error (RMSE RPE) [Sturm et al., 2012], averaged over 10 runs with randomised robot motions in Figure 6.3, for varying numbers of robots in the area. Both ATE and RPE are computed over the full trajectory. We see that GBP and GTSAM have similar ATE across all evaluations, with only a small loss of accuracy for GBP when the number of robots is low.

Table 6.1: The RMSE ATE of the trajectories for different numbers of robots in simulation. We report the mean error and the standard deviation of 10 runs with different random initialisation.

N	Range [m]	Noise [m, rad]	GTSAM $\mu \pm \sigma$ [m]	GBP $\mu \pm \sigma$ [m]	Windowed $\mu \pm \sigma$ [m]
16	10	0.01, 0.05	0.660 ± 0.217	0.770 ± 0.183	0.934 ± 0.152
	10	0.05, 0.1	0.690 ± 0.218	0.773 ± 0.192	0.975 ± 0.127
	30	0.01, 0.05	0.063 ± 0.024	0.066 ± 0.025	0.088 ± 0.042
	30	0.05, 0.1	0.081 ± 0.019	0.087 ± 0.021	0.117 ± 0.040
32	10	0.01, 0.05	0.314 ± 0.062	0.462 ± 0.080	0.561 ± 0.076
	10	0.05, 0.1	0.344 ± 0.049	0.437 ± 0.058	0.597 ± 0.063
	30	0.01, 0.05	0.015 ± 0.003	0.016 ± 0.003	0.022 ± 0.008
	30	0.05, 0.1	0.035 ± 0.003	0.036 ± 0.004	0.043 ± 0.006
64	10	0.01, 0.05	0.154 ± 0.018	0.290 ± 0.072	0.358 ± 0.072
	10	0.05, 0.1	0.181 ± 0.023	0.256 ± 0.064	0.375 ± 0.080
	30	0.01, 0.05	0.009 ± 0.001	0.009 ± 0.001	0.010 ± 0.003
	30	0.05, 0.1	0.023 ± 0.001	0.023 ± 0.001	0.024 ± 0.003
128	10	0.01, 0.05	0.060 ± 0.004	0.105 ± 0.016	0.134 ± 0.015
	10	0.05, 0.1	0.082 ± 0.004	0.102 ± 0.009	0.158 ± 0.011
	30	0.01, 0.05	0.006 ± 0.000	0.006 ± 0.000	0.006 ± 0.000
	30	0.05, 0.1	0.016 ± 0.000	0.016 ± 0.000	0.016 ± 0.000

As the number of robots increases, the difference in ATE across the different approaches becomes negligible. Windowed GBP reaches comparable accuracy to GTSAM and GBP, even with a significantly smaller computational cost when compared to full GBP optimisation.

A similar pattern is observed when we vary the sensor noise and range. As we increase the number of robots, the difference in error across different approaches reduces. The sensor noise is increased to $\sigma_r = 0.05$m, $\sigma_b = 0.1$ rad, and the range is limited to 10m. Tables 6.1 to 6.3 summarises the result of sweeps over the different parameters.

When the sensor range is limited or the number of robots is small, fewer observations are made and robots will drift more from their correct trajectories. Being a local algorithm, GBP can rapidly optimise local, high-frequency component errors in the network [Davison and Ortiz, 2019], while it requires more iterations for information to propagate across the graph to optimise lower frequency component errors, such as longer drifts. This property is observable in Figure 6.3 where the difference in ATE between GTSAM and GBP for 16 robots is noticeable due to the low-frequency noise. Since the range of the sensors is limited, fewer observations are made when the total number of robots is small, as the arena has a lower density of robots. Similarly, as shown in Table 6.1 for N=128. When the sensor range is 30m, GTSAM and GBP achieve the same ATE. However, when the sensor range is 10m, a small difference exists. However; in both cases for the relative metric RPE, similar performance is achieved even with a small number of robots, demonstrating that the local/high-frequency component errors are correctly smoothened.

Increasing the number of iterations improves convergence as more messages are exchanged. We

Table 6.2: The translational RMSE RPE of the trajectories for different numbers of robots in simulation. We report the mean error and the standard deviation of 10 runs with different random initialisation.

N	Range [m]	Noise [m, rad]	GTSAM $\mu \pm \sigma$ [m]	GBP $\mu \pm \sigma$ [m]	Windowed $\mu \pm \sigma$ [m]
16	10	0.01, 0.05	0.321 ± 0.084	0.349 ± 0.071	0.382 ± 0.064
	10	0.05, 0.1	0.336 ± 0.082	0.356 ± 0.077	0.396 ± 0.064
	30	0.01, 0.05	0.087 ± 0.020	0.087 ± 0.019	0.090 ± 0.018
	30	0.05, 0.1	0.119 ± 0.025	0.118 ± 0.025	0.120 ± 0.025
32	10	0.01, 0.05	0.207 ± 0.028	0.235 ± 0.041	0.264 ± 0.048
	10	0.05, 0.1	0.227 ± 0.032	0.238 ± 0.034	0.276 ± 0.039
	30	0.01, 0.05	0.068 ± 0.015	0.068 ± 0.016	0.069 ± 0.016
	30	0.05, 0.1	0.102 ± 0.021	0.101 ± 0.021	0.102 ± 0.020
64	10	0.01, 0.05	0.164 ± 0.015	0.198 ± 0.031	0.216 ± 0.033
	10	0.05, 0.1	0.191 ± 0.015	0.197 ± 0.020	0.227 ± 0.030
	30	0.01, 0.05	0.054 ± 0.007	0.053 ± 0.006	0.053 ± 0.006
	30	0.05, 0.1	0.078 ± 0.009	0.077 ± 0.009	0.077 ± 0.009
128	10	0.01, 0.05	0.092 ± 0.009	0.105 ± 0.009	0.109 ± 0.009
	10	0.05, 0.1	0.120 ± 0.004	0.126 ± 0.003	0.135 ± 0.005
	30	0.01, 0.05	0.047 ± 0.002	0.047 ± 0.002	0.047 ± 0.002
	30	0.05, 0.1	0.068 ± 0.003	0.068 ± 0.003	0.068 ± 0.003

can verify this in Figure 6.4, where we vary the number of iterations per step between 10-50. We use 50 robots and report the average over 50 different runs. We see that as the number of iterations per step increases, ATE decreases; however, with diminishing returns. The optimal number of iterations per step depends on many factors (*e.g.* topology of the graph, communication pattern) and is an interesting direction for further research.

6.5.3 Operation with a Large Number of Agents

In terms of computational performance, it would not be meaningful to report the speed of our C++ CPU simulation of the Robot Web algorithm, which is designed to be fully distributed across a large number of devices. However, in fact, our simulation can run in real-time on a laptop for problems involving 100 robots or beyond using Windowed GBP, in particular, because it is designed to take advantage of CPU parallelism using OpenMP.

Instead, we present an experiment which demonstrates the scaling properties of Windowed GBP in a mode where the computation and communication work *per robot* is bounded. Figure 6.5 shows the average ATE of all robot pose estimates as the number of interacting robots in our arena is raised from 32 to 1024. The result is an average of over 10 different runs. Each robot measures nearby robots but is allowed to communicate with *one other robot* sampled based on Equation 6.4 per GBP iteration. Robot Web handles this extreme packing and scaling straightforwardly, and the ATE for all robots continues to decrease as robots are added due to the favourable high inter-

Table 6.3: The rotational RMSE RPE of the trajectories for different numbers of robots in simulation. We report the mean error and the standard deviation of 10 runs with different random initialisation.

N	Range [m]	Noise [m, rad]	GTSAM $\mu \pm \sigma$ [deg]	GBP $\mu \pm \sigma$ [deg]	Windowed $\mu \pm \sigma$ [deg]
16	10	0.01, 0.05	0.626 ± 0.028	0.642 ± 0.025	0.776 ± 0.036
	10	0.05, 0.1	0.639 ± 0.031	0.650 ± 0.030	0.789 ± 0.033
	30	0.01, 0.05	0.510 ± 0.012	0.510 ± 0.012	0.528 ± 0.019
	30	0.05, 0.1	0.535 ± 0.012	0.535 ± 0.013	0.579 ± 0.015
32	10	0.01, 0.05	0.574 ± 0.011	0.592 ± 0.015	0.732 ± 0.034
	10	0.05, 0.1	0.589 ± 0.011	0.596 ± 0.012	0.750 ± 0.019
	30	0.01, 0.05	0.496 ± 0.007	0.496 ± 0.007	0.501 ± 0.007
	30	0.05, 0.1	0.519 ± 0.007	0.519 ± 0.007	0.545 ± 0.009
64	10	0.01, 0.05	0.549 ± 0.007	0.567 ± 0.013	0.695 ± 0.034
	10	0.05, 0.1	0.569 ± 0.009	0.573 ± 0.008	0.711 ± 0.029
	30	0.01, 0.05	0.475 ± 0.003	0.476 ± 0.003	0.477 ± 0.004
	30	0.05, 0.1	0.506 ± 0.003	0.506 ± 0.003	0.517 ± 0.003
128	10	0.01, 0.05	0.518 ± 0.004	0.523 ± 0.004	0.576 ± 0.006
	10	0.05, 0.1	0.540 ± 0.004	0.542 ± 0.004	0.614 ± 0.006
	30	0.01, 0.05	0.443 ± 0.004	0.443 ± 0.004	0.444 ± 0.004
	30	0.05, 0.1	0.490 ± 0.002	0.490 ± 0.002	0.495 ± 0.002

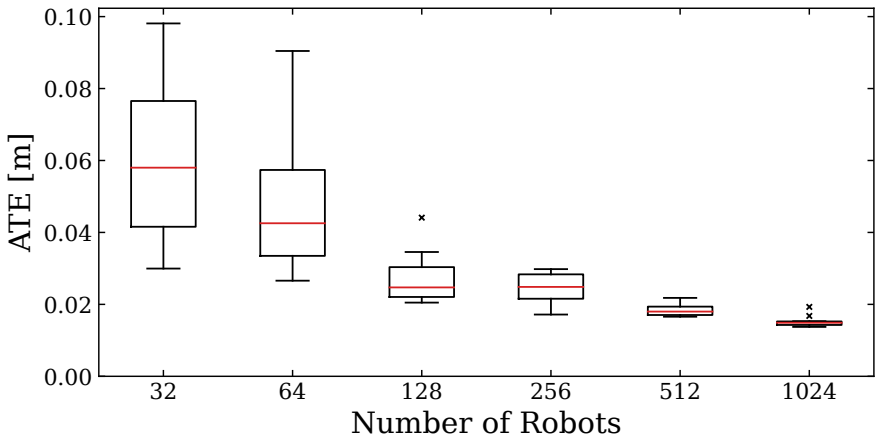


Figure 6.5: Extreme scaling: in a simulated environment, we increase the number of robots in the arena to over 1000, with each robot communicating with only one other per iteration of Windowed GBP, and therefore having a per-robot bounded computation and communication workload. The average ATE in all robots' poses continues to decrease as we increase the number of robots. Each result is a summary of 10 runs with different random initialisation.

connectedness of the whole graph, despite the minimal communication allowed. These results indicate the true potential of Robot Web methods towards very high numbers of simple interacting devices.

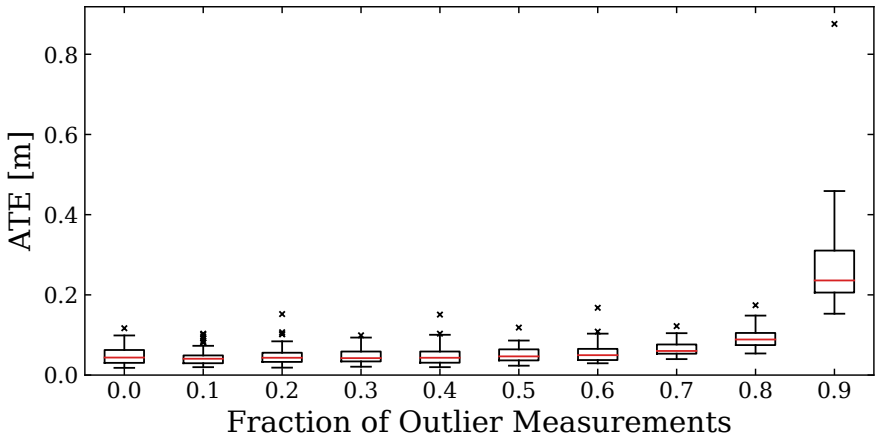


Figure 6.6: Robust factors enable remarkable resilience to a large fraction of outlier inter-robot sensor measurements, with ATE remaining low up to 70–80% of corrupt measurements to which a large amount of uniform noise is added. In a simulated environment, 50 robots are moving in an environment with 10 known landmarks for 100 poses each. Each result is a summary of 50 runs with different random initialisation.

6.5.4 Operation with Outlier Measurements and Robust Factors

Here, we demonstrate the robustness of GBP using the method for handling robust factors from FutureMapping 2 [Davison and Ortiz, 2019] and the robust kernel from DCS [Agarwal et al., 2012]. 50 robots are used, each with a sliding window of 5. In Figure 6.6, we show what happens to the ATE when we increase the fraction of the range-bearing measurements containing the uniform noise. We see that a huge fraction of up to 80% of measurements can be completely corrupted but still handled by the robust measurement kernel with very little effect on the overall accuracy of the network. This again shows the advantage of the heavily inter-connected network which GBP allows us to efficiently and incrementally optimise in a distributed manner. In this network, each pose estimate is highly over-constrained, and this is what allows the robust kernel to weed out outlier measurements.

6.5.5 Operation with Unreliable Communication

In multi-robot systems, another potential problem is the reliability of the communications. Often robots will communicate with best-effort, meaning messages can get lost in the network. Robustness against such data loss can be challenging; however, GBP is not significantly affected, as the message scheduling can be random. Here, we imagine that data transmission is quantised at the level of individual messages, as it might be with certain types of communication technology, and experiment to see the effect of the loss of a random fraction of messages between robots.

In Figure 6.7, we force the network to drop the messages randomly with a fixed probability which we gradually increase and investigate how that affects ATE. For example, if Robot α sends 3 rows of message $\{M_1, M_2, M_3\}$ to Robot β, the network may drop M_2, and Robot β will only receive

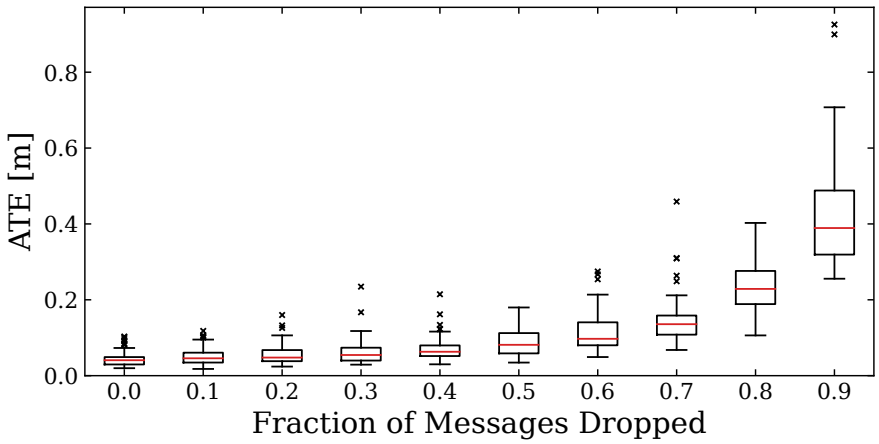


Figure 6.7: Robot Web is highly robust to a high fraction of randomly dropped messages. In a simulated environment, 50 robots are moving in an environment with 10 known landmarks for 100 poses each. The result is a summary of 50 runs with different initialisation.

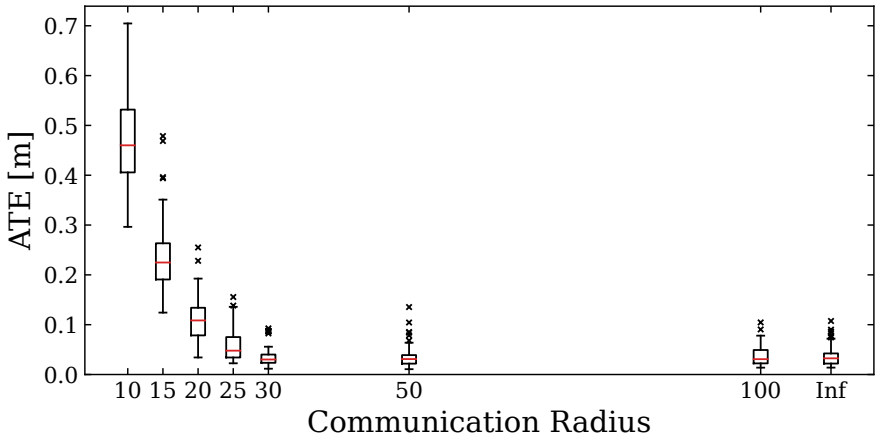


Figure 6.8: Analysis of the effect of varying the allowed communication range. 'Inf' means all robots are allowed to communicate with any other robot. While increasing the communication radius improves the performance, 30m onwards, the difference is negligible. In a simulated environment, 50 robots are moving in an environment with 10 known landmarks for 100 poses each. The result is a summary of 50 runs with different initialisation.

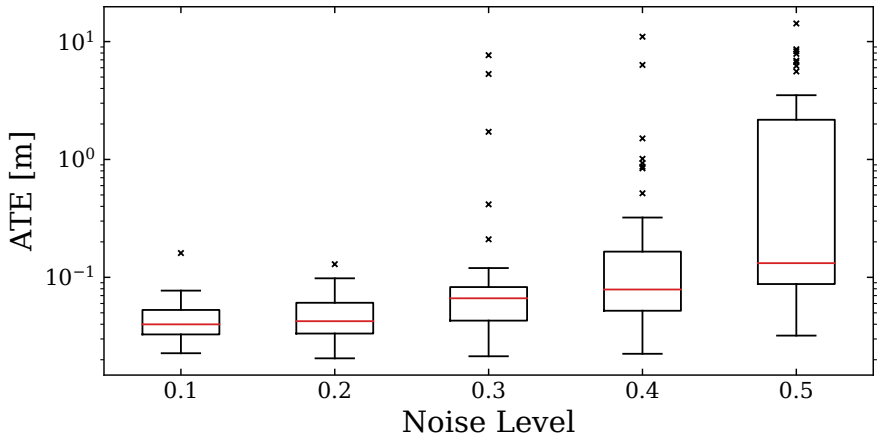


Figure 6.9: Robot Web demonstrates its resilience to large initialisation errors. We add to the initial pose a noise sampled from a Gaussian with a standard deviation of (n m, n m, n rad), where n represents the noise level. Note that the graph is plotted on a logarithmic scale. In a simulated environment, 50 robots are moving in an environment with 10 known landmarks for 100 poses each. The result is a summary of 50 runs with different initialisation.

$\{M_1, M_3\}$. In this experiment, we also see very advantageous properties for GBP, which retains a low ATE up to at least 50% message loss in this setting.

We further evaluate the effect of poor communication in terms of communication range. The communication radius of the robots was adjusted to range from 10m to 100m and an infinite radius. In line with previous experiments, each robot communicates with only one other robot per iteration. We disable the sliding window and perform a full pose update for this evaluation such that robots can exchange messages asynchronously on rendezvous.

As shown in Figure 6.8, reducing the communication range decreases the performance; however, beyond a radius of 25m, the performance improvements are minimal. In this configuration, there may be robots who never communicate with one another though they've made measurements of each other. As GBP has no synchronisation, such cases are simply ignored without the need for specific procedures. The asynchronous communications; however, lead to inconsistencies in the linearisation points across multiple robots which potentially leads to poor convergence. However, as shown in Figure 6.8, the performance gap between 25m radius and beyond is negligible, indicating robustness against these inconsistencies.

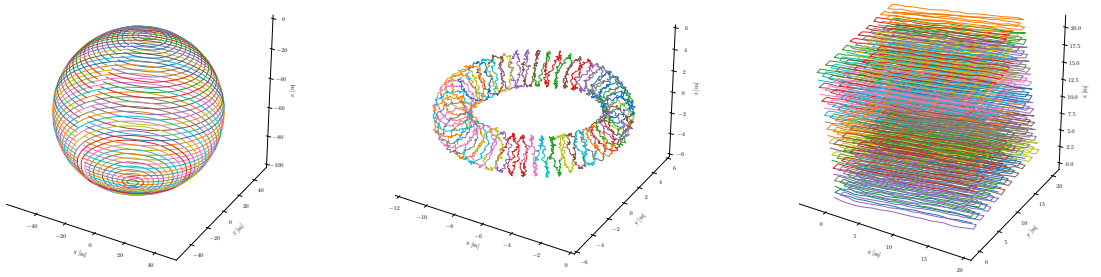
6.5.6 Operation Under Poor Initialisation

The initialisation is important for multi-robot localisation, especially for handling outlying measurements. However, good initialisation may not always be available in the real world. Here, we analyse the effect of increasing the noise on the initialisation and when the system breaks. We vary the $(\sigma_x, \sigma_y, \sigma_\theta)$ from (0.1m, 0.1m, 0.1 rad) to (0.5m, 0.5m, 0.5 rad). The percentage of outlier range-bearing sensor measurements remains to be fixed at 10%.

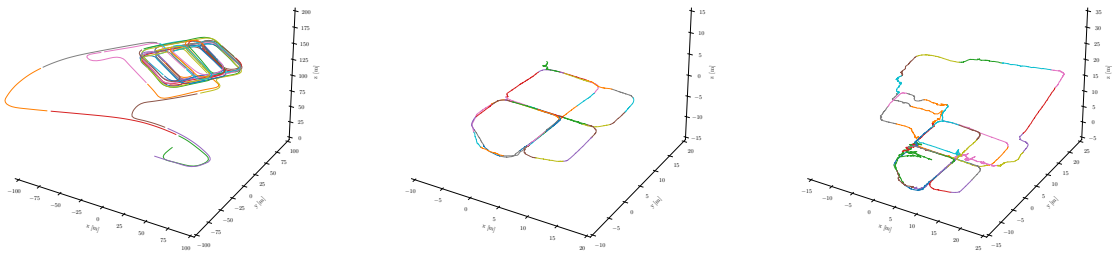
Table 6.4: A comparison of the different distributed solvers for solving multi-robot pose-graph optimisation. We report the initial cost, the solution of centralised Gauss-Newton (GN), and the cost and the number of iterations required for convergence for the different distributed solvers: distributed Block Gauss-Seidel (DGS) and distributed Block Jacobi Method from [Choudhary et al., 2017] and ours. Across all datasets, though distributed, our method and DGS obtains similar cost to the centralised GN.

Dataset	Initial Cost	Centralised GN	Block Gauss-Seidel		Block Jacobi Method		Ours	
			#Iter	Cost	#Iter	Cost	#Iter	Cost
Sphere	1.28863×10^6	8.43504×10^2	723	8.52218×10^2	10000	3.28738×10^3	1240	8.58949×10^2
Torus	1.88612×10^6	1.21137×10^4	847	1.23950×10^4	6964	1.25181×10^5	1495	1.22184×10^4
Parking Garage	8.36192×10^3	6.31262×10^{-1}	117	7.93764×10^{-1}	5142	8.16846×10^3	1472	6.94700×10^{-1}
Cubicle	2.53917×10^6	3.18310×10^2	701	3.38483×10^2	9709	2.20025×10^3	244	3.97225×10^2
Rim	4.06073×10^7	1.24992×10^3	2355	6.50345×10^3	6142	1.00088×10^{23}	2932	3.60934×10^3
Grid	7.21751×10^7	4.21596×10^4	327	4.24620×10^4	5613	4.96610×10^4	1608	4.23358×10^4

We plot the graph on a logarithmic scale for clarity but notice that at noise level (0.5m, 0.5m, 0.5 rad), the value of the upper whisker is 3.51m whereas at (0.4m, 0.4m, 0.4 rad) it is 0.32m, clearly showing that the error explodes. Initialisation is critical for outlier rejection, and with a poor initialisation, good observations will have high energy and possibly lie in the outlier region of the robust kernel, making the optimisation problem challenging. While our approach demonstrates robustness against up to a large initialisation noise of (0.2m, 0.2m, 0.2 rad), improving the robustness to poor initialisation is an interesting direction for future works.



(a) Sphere (b) Torus (c) Cube



(d) Garage (e) Cubicle (f) Rim

Figure 6.10: Final trajectory returned by GBP on benchmark datasets [Carlone et al., 2015a]. Here, the trajectories are split among 50 robots.

6.5.7 Joining and Leaving the Robot Web

The Robot Web is fully dynamic because each robot does not need any information about the group as a whole, so robots can join or leave freely. When new robots are added, randomly into the arena, it is initialised at the centre of the arena and starts to participate in the Robot Web. It does not start to move until it believes that it has a good pose estimate. This decision is based on each robot monitoring the robust scaling of its factors, which is based on the Mahalanobis distance. Our implementation checks whether (a) the average scaling for all outgoing factors is > 0.95, and (b) that there are at least 8 different observations. Until these criteria are met, the newly-added robots send empty messages on the inter-robot factors and therefore do not affect the already-initialised robots until they are confident enough to start moving and properly taking part in the web. A video demonstration of this in simulation is available here:
https://rmurai.co.uk/projects/RobotWeb#dynamic

6.5.8 Comparison against other solvers

While the focus of the work is on distributed localisation using range-bearing sensor measurements, the fact that our method operates on an arbitrary factor graph enables the framework to work with different sensor modalities, for instance, inter-robot $\mathbf{SE}(3)$ transformation, often used in distributed pose-graph optimisation (PGO). Here, we solve the following problem:

$$\min_{\substack{\mathbf{t}_i \in \mathbb{R}^3, \\ \mathbf{R}_i \in \mathbf{SO}(3), \forall i}} \frac{1}{2} \sum_{\{i,j\} \in \varepsilon} \tau_{ij} \|\mathbf{t}_j - \mathbf{t}_i - \mathbf{R}_i \mathbf{t}^z_{i,j}\|_2^2 + \kappa_{ij} \|\mathbf{R}_j - \mathbf{R}_i \mathbf{R}^z_{i,j}\|_F^2 \,, \tag{6.5}$$

where ε is a set of all measurements, $\mathbf{R}_i$ is a rotation variable, $\mathbf{t}_i$ is a translation variable, $\mathbf{R}^z_{i,j}$ is the measured rotation from i to j and similarly $\mathbf{t}^z_{i,j}$ is the measured translation from i to j. τ_{ij}, κ_{ij} are the noise parameter computed from the dataset as done in [Rosen et al., 2019, Tian et al., 2020, Fan and Murphey, 2020].

The main complexities of PGO lie in how we handle poor initialisation. As the optimisation problem is non-convex, there exist many local minima. If we directly solve Equation 6.5, we will get stuck in a local minimum, even with small noise [Carlone et al., 2015b]. Following [Choudhary et al., 2017], we thus solve a relaxed, linear problem in two stages in a distributed manner. First, we solve the rotation problem:

$$\min_{\mathbf{R}_i \in \mathbf{SO}(3), \forall i} \frac{1}{2} \sum_{\{i,j\} \in \varepsilon} \kappa_{ij} \|\mathbf{R}_j - \mathbf{R}_i \mathbf{R}^z_{i,j}\|_F^2 \,. \tag{6.6}$$

We solve the quadratic relaxation of this problem, by dropping the $\mathbf{SO}(3)$ constraint and then projecting the solution back to $\mathbf{SO}(3)$ via SVD.

We then solve for the full pose using a linear approximation of rotation perturbation:

$$\begin{aligned} \min_{\mathbf{t}_i, \theta_i \in \mathbb{R}^3, \forall i} \frac{1}{2} \sum_{\{i,j\} \in \varepsilon} & \tau_{ij} \|\mathbf{t}_j - \mathbf{t}_i - \mathbf{R}_i \tilde{\mathrm{Exp}}(\theta_i) \mathbf{t}^z_{i,j}\|_2^2 \\ & + \kappa_{ij} \|\mathbf{R}_j \tilde{\mathrm{Exp}}(\theta_j) - \mathbf{R}_i \tilde{\mathrm{Exp}}(\theta_i) \mathbf{R}^z_{i,j}\|_F^2 \,, \end{aligned} \tag{6.7}$$

where $\tilde{\text{Exp}}(\theta) = \mathbf{I}_3 + \mathbf{S}(\theta)$, and $\mathbf{S}(\theta)$ is a skew symmetric matrix.

In [Choudhary et al., 2017], Equation 6.6 and Equation 6.7 is solved using distributed Block Gauss-Seidel (DGS) or distributed Block Jacobi method. Here, we compare GBP and DGS for solving the two-stage PGO problem. We compare against DGS as it is used as an initialisation for other works [Tian et al., 2020, Tian et al., 2021], and relaxation is simple to perform with the factor graph framework. In our evaluation, we report the initial and final cost and the number of iterations required to satisfy the termination condition. We evaluate the trajectories on pose-graph optimisation dataset [Carlone et al., 2015b]. Each trajectory is split into 50 segments to simulate a multi-robot pose-graph.

The setup of our evaluation favours the DGS. We count one iteration as a full DGS sweep, where the robots sequentially send the updated information to the next robots in a specific order. GBP, on the contrary is robot-wise parallel and does not require coordinated updates. Hence, the communication pattern of GBP is closer to the distributed Block Jacobi method rather than DGS. Furthermore, we enable flagged initialisation for both DGS and distributed Block Jacobi method. For all methods, we terminate the iterations once the norm of the change in the rotation or the pose is below a specified threshold. Here, we use 10^{-2} as the threshold for both the rotation and the pose update, for all of our distributed solvers as recommended in [Choudhary et al., 2017]. Furthermore, for DGS and distributed Block Jacobi, we use the recommended relaxation parameter of 1.0 which we too found to work the best. We allow all the solvers to run for up to 10000 iterations.

As shown in Table 6.4, GBP performs comparable to DGS and obtains cost close to the centralised Gauss-Netwon solver, though the setup favours DGS, and DGS has distributed pose-graph specific heuristics such as flagged initialisation. Compared to the distributed Block Jacobi method which has a similar communication pattern as GBP, GBP performs significantly better, both in terms of final cost and the number of iterations. This result highlights the generality of GBP and makes GBP a promising alternative to the existing distributed solvers. Devising a fair and complete evaluation of different distributed solvers is an interesting direction for future work.

6.6 Demonstrations and Experiments in a Real-World

To provide concrete evidence of the effectiveness of our approach, we have evaluated the real robots running our system on onboard devices in a distributed manner.

6.6.1 Evaluation Setup

To evaluate our approach with real robots, we used nine TurtleBot3 Burgers as the robot platform. The robots (as shown in Figure 6.11) were equipped with a Raspberry Pi 3B+ computer with a Cortex-A53 64-bit 1.4GHz processor and 1GB of RAM as the onboard computer. In addition, each robot was fitted with an AprilTag-labeled cube – with the same tag on all sides – and an

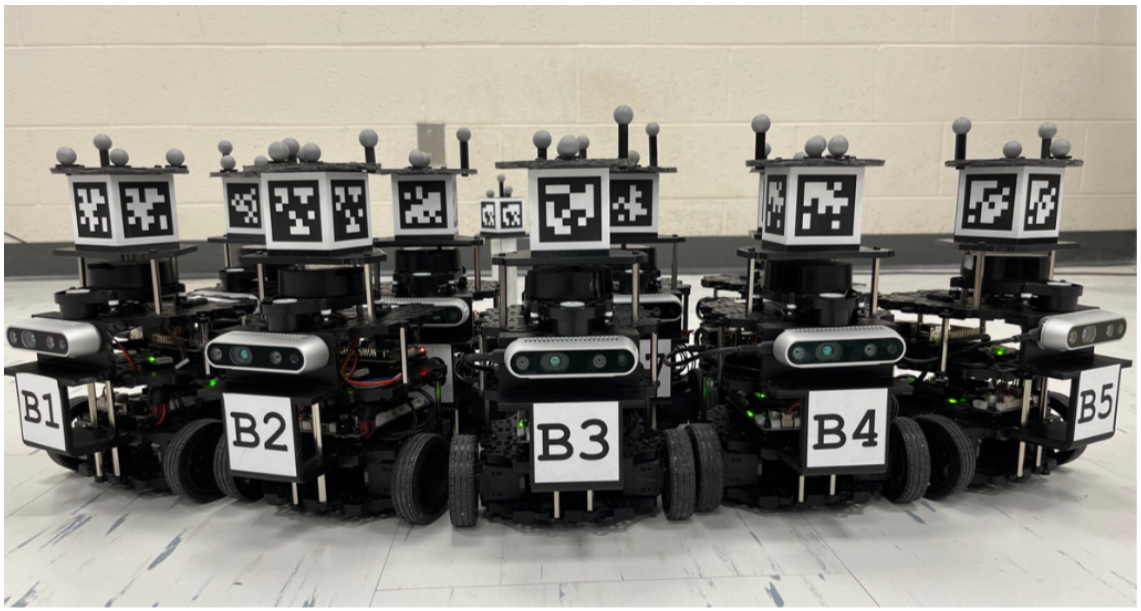
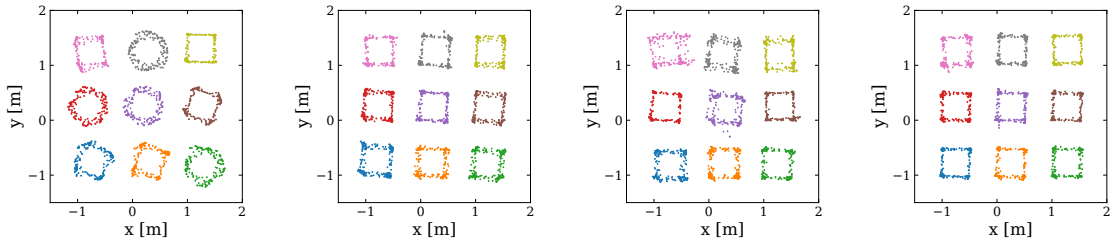


Figure 6.11: Image of a Turtlebot3 Burger used in the real robot experiment. It is fitted with AprilTag-labelled cubes, an Intel-Realsense D435i camera and Vicon markers. Vicon markers are only used to obtain the ground-truth trajectories, used for the evaluation. The depth image, laser scanner and IMU are not used in any of the experiments.

Intel-Realsense D435i camera. The RGB images captured by the camera and the data from the wheel encoder served as the sensory input. To simplify the setup, the depth image, IMU, and laser scanner were disabled, and for the odometry, only the wheel odometry was used. Each robot had knowledge of the size and location of the AprilTag [Olson, 2011], the camera position, and the calibration parameters. As we have many robots, factory calibration was used for the odometry and the camera. This is unideal as it adds systematic bias; however, our approach was still able to function effectively.

During the described experiment, the robots are instructed to follow a square trajectory. When the Robot Web system detects a drift in the robot's position, a heuristic is used to correct the pose. For the drift of less than 5cm, proportional control is applied to bring the robot back onto the desired trajectory. Otherwise, the robot turns to face the next corner of the trajectory to correct the pose.

All computation, including GBP optimisation, pose correction, and inter-device communication via ROS2, runs on the onboard computer, highlighting the computational efficiency of our approach. We assume that the robots know the mapping between unique IDs and IP addresses in advance and that there is a shared/synchronised clock for all observations. When an AprilTag is detected, observation is transformed into a range-bearing measurement. We are unable to obtain relative transformation measurements – which include both translation and rotation – as the same AprilTag is used on all sides of the cube, so the orientation is ambiguous.



(a) No landmarks, no inter-device communication.

(b) No landmarks, with inter-device communication.

(c) With landmarks, no inter-device communication.

(d) With landmarks, with inter-device communication.

Figure 6.12: Nine real robots are moving in a square trajectory, and the motion captured by the Vicon system is plotted. It is clearly visible that using inter-device communication improves the localisation accuracy.

6.6.2 Implementation Details

We use ROS2 Foxy [Macenski et al., 2022] for all the robots. The Publish-subscribe model, as described in Section 6.4.2 is used for message passing. In ROS2 this entails simply subscribing to the topics (*e.g.* for robot 1, it will subscribe to `robot_1/variable_msg`, `robot_1/factor_msg`) and publishing to either the variable/factor of other robots along the inter-device factor.

GBP runs on its own thread, and the passing of internal messages runs as fast as possible. GBP process is interleaved with the subscriber which receives the inter-device messages. For simplicity, a single coarse lock is used to avoid concurrency problems (adding inter-device messages to the internal factor graph); however, as all update operations of GBP are local, it is possible to use a finer lock. The publisher runs at 10Hz, publishing the outgoing messages. Best-effort delivery is used; hence, there is no delivery guarantee. We emphasise that the publishing and receiving of the messages are not synchronised, and robots receive messages at arbitrary timings (potentially out of order).

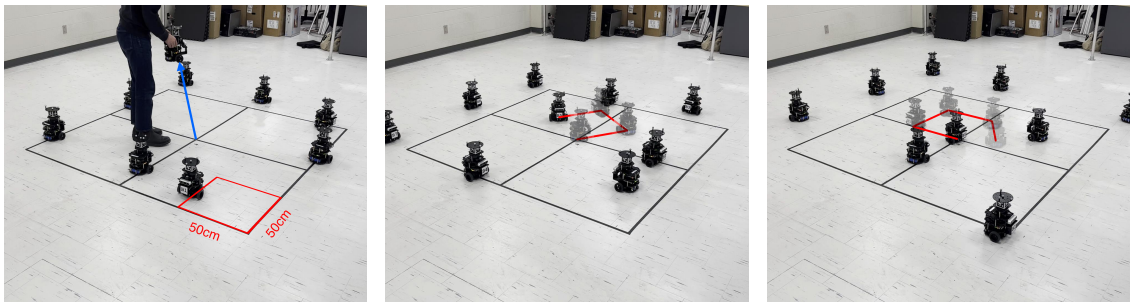
We set the sensor noise to be: $\sigma_x = 0.01$m, $\sigma_y = 0.01$m, and $\sigma_\theta = 1^\circ$ for the prior; $\sigma_x = 0.01$m, $\sigma_y = 0.005$m, and $\sigma_\theta = 1^\circ$ for the odometry; and $\sigma_b = 0.01$m, and $\sigma_\theta = 1^\circ$ for the range-bearing. All robots run GBP with a window size of 5. DSC [Agarwal et al., 2012] is used for the robust kernel with $\Phi = 10$. The variable nodes are after any forward motion or a rotation, and in all the experiments, 75 poses per robot were added to the graph.

6.6.3 Multi-Robot Localisation Evaluation

In this section, we evaluate the localisation accuracy of our approach. We evaluate under two different settings, with and without landmarks. Four landmarks are used, and their position is known to the robots in advance. Figure 6.12 shows the trajectory captured by the Vicon motion capture system. In all runs, robots are moving for 10 minutes. It is clear that Robot Web localises the robots well and allows them to operate for a long period without drifting.

Table 6.5: The table below shows the RMSE ATE of the real robot experiment. The RMSE ATE of the real robots is computed against the observations made by the Vicon motion capture system. The table summarises the impact of inter-device communication and the availability of landmarks on the RMSE ATE. We report the mean error and the standard deviation of the nine robots.

Communication	Landmark	$\mu \pm \sigma$ [m]
False	False	0.162 ± 0.085
True	False	$\mathbf{0.043 \pm 0.020}$
False	True	0.071 ± 0.020
True	True	$\mathbf{0.028 \pm 0.007}$



(a) (b) (c)

Figure 6.13: Here, nine robots are running Robot Web. Each robot starts on the vertices of the grid on the floor and moves in a square pattern (50cm x 50cm). In (a) during operation, one robot is removed from the system (*e.g.* for maintenance) and then added back with an incorrect pose. As a result, the robot fails to follow the square pattern, as shown in (b). However, using Robot Web, the robot is able to successfully relocalise, as shown in (c), and returns to following the square trajectory.

The RMSE ATE of the real robots is computed against the observations made by the Vicon motion capture system. The result is summarised in Table 6.5. As expected, whether there are landmarks or not, using inter-device communication, *i.e.* Robot Web, improves the accuracy of localisation. The use of sparse landmarks is insufficient for good localisation without inter-device communication. This is clear both qualitatively by comparing (b) and (c) of Figure 6.12, and quantitatively in Table 6.5 by comparing: no landmarks, with inter-device communication; and with landmarks, no inter-device communication.

6.6.4 Relocalisation Demonstration

In a multi-robot system, there are many potential sources of failure for the robots. For instance, a robot might need to be stopped for maintenance due to a low battery, or it could be accidentally bumped out of position by a person. These types of external influences are often non-Gaussian, and if the system only accounts for Gaussian noise, it will not be able to accurately handle these unexpected events.

In Robot Web, while GBP assumes a Gaussian noise, robust factors allow the system to handle

non-Gaussian noise as well. In Figure 6.13, we lift a moving robot and place it back in the wrong position. This disorients the robot, and it is unable to follow the square trajectory. However, after a few observations, the robot relocalises and returns to follow the square trajectory. During this relocalisation process, other robots are unaffected by the wrongly positioned robot as the robust factor heavily down-weights its influence until the wrongly positioned robot is correctly localised. Due to the error between the position of the robot and its estimate, the measurements made of this robot by the others will have high residuals and thus will be down-weighted by the robust kernel. A video of the relocalisation demo is available here:
https://rmurai.co.uk/projects/RobotWeb#reloc

6.7 Ongoing Research Topics

We have demonstrated the essential operation of the Robot Web both in a simulation and in a real-world, truly distributed implementation on multiple robots. The properties of the method are extremely promising, and here we discuss some important research directions going forward.

6.7.1 More General Parameterisation

Our current implementation makes several simplifying assumptions, but we believe that all of these are fairly straightforward to remove within the Robot Web framework with some further work.

- We currently assume that inter-robot measurement factors, stored by the robot with the sensor, always correspond to observations of the position of the centre of the second robot. This would already allow a practical implementation for 2D planar robots which each carry a single observable beacon above their centres. More realistically, each robot might have several or many observable features, and these will be located on any point on its structure. We can deal with this by adding additional internal variables to the second robot, connected to its main pose by 'perfect' factors, representing the positions of the observable features relative to its body, with positions that only need to be known to the second robot.

- Our current assumption that all robots have pose variables defined at the same rate and at corresponding times could be relaxed by measurement factors which connect to multiple variables at the receiving robot and interpolate the measurement between poses.

We might take the Robot Web idea even further to also apply *inside* a single robot's modular body. The different parts, actuators and sensors that make up the robot might use Web interfaces between them to enable distributed joint estimation and very general modularity.

6.7.2 Efficient Long-Term Operation

If we keep the full history of all pose variables for each robot, and all measurement factors, eventually the computation, storage and communication capacity of each robot would become overloaded. Of these, inter-robot communication is likely to be the main bottleneck. We showed one simple approach to dealing with this via time windowing, where poses older than a threshold are discarded, and this gives good performance when robots have bounded drift due to the presence of known beacons.

A more general approach to bounding the growth of the graph could be based on incremental abstraction [Ortiz et al., 2022], where past variables and factors are not deleted but grouped into more efficient blocks with minimal loss of accuracy. For instance, a set of well-estimated pose variables from the past could be grouped into an abstract trajectory segment, represented by far fewer variables. Factors could also be grouped. Achieving this incremental abstraction in a fully distributed way across multiple robots however will require substantial research.

6.8 Discussion and Conclusions

We have presented a method for distributed multi-robot localisation in the context of a larger 'Robot Web' vision for how heterogeneous groups of intelligent robots and devices of the future could cooperate and coordinate. This approach could be important at a time when many different companies and organisations are building spatially aware devices, and offers a distributed, interoperable alternative to a single unified cloud maps solution.

As the performance and scale of many-robot systems may greatly improve due to work such as ours, it is important to consider potential ethical concerns. A robust, large-scale robot group or 'swarm' has many possible positive applications, such as the automation of farming or environmental surveillance via many low-cost devices, which could be much more efficient overall than a small number of large devices. However, there are possible ethical concerns with swarms of autonomous, weaponised drones

We believe that our work overall could indicate a positive direction for the operation of distributed multi-robot systems via the specification that the Robot Web allows and demands of an **open communication protocol**. If the majority of the moving intelligent devices were to take part in such a system by publishing and reading localisation messages via this open protocol, it would be greatly to the advantage of any newly built devices to also take part, to exchange open messages, and to benefit from the system. This would mean that the whole system might work in a way similar to the World Wide Web, and some degree of global control would be possible via the interpretability of the protocol and perhaps more specific safety measures built into it. We believe that it is better for devices to be exchanging clearly interpretable geometric information than cryptic coded messages (as would emerge for instance in a possible distributed 'graph neural network' system for localisation, where the format of messages is learned rather than designed — and we should

add here our view that a learned alternative to our method is also likely to be far less flexible and efficient).

These are ongoing issues to be debated as the technology advances, and we as authors believe that researchers should openly engage with these issues and play a part in designing the correct principles into the technology.

GBP is a local optimiser and combined with the overconfidence problem in a loopy graph, it struggles to remove the low-frequency error. To accelerate convergence, hierarchical optimisation such as multi-grid methods or batching many small factors into a single large factor would be an interesting direction.

In the longer term future, the distributed coordination of intelligent moving systems is a key part of the concept of 'intelligent matter', where distribution and communication might be at the microscopic level to enable new classes of technology such as micromachines [Huang et al., 2022] which can self-organise in ways that might approach the capabilities of biological systems [Kaspar et al., 2021]. Recently, it has been shown that essentially the same computation framework that we have demonstrated in Robot Web using GBP can also be applied to multi-robot motion planning [Patwardhan et al., 2023]. Efficient, robust distributed localisation will be one of the most important enabling layers of such systems.

CHAPTER 7

Distributed Simultaneous Localisation and Auto-Calibration using Gaussian Belief Propagation

Contents

We present a novel scalable, fully distributed, and online method for simultaneous localisation and extrinsic calibration for multi-robot setups. Individual a priori unknown robot poses are probabilistically inferred as robots sense each other while simultaneously calibrating extrinsic parameters of their sensors and markers using Gaussian Belief Propagation. In the presented experiments, we show how our method not only yields accurate robot localisation and auto-calibration but also is able to perform under challenging circumstances such as highly noisy measurements, significant communication failures or limited communication range.

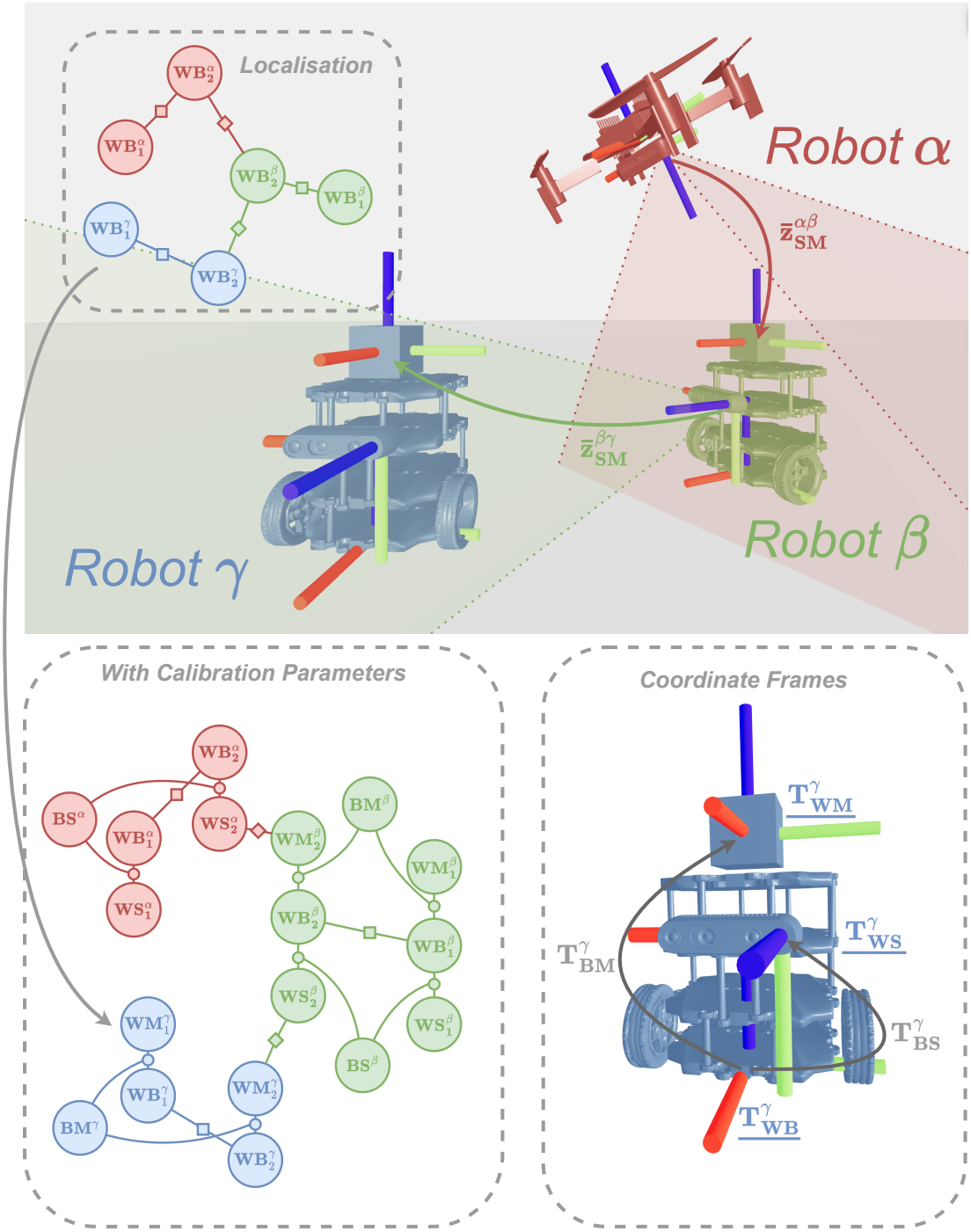


Figure 7.1: Overview of the proposed auto-calibrating localisation system for three heterogeneous robots (top). Each robot observes the markers M placed on its peers to establish measurement $\bar{\mathbf{z}}_{SM}$ using sensor S mounted on top of a moving base B. Using the proposed methodology, the robots' relative positions and their calibration parameters are retrieved in a distributed and asynchronous fashion performing probabilistic inference on a factor graph. We refer to $\mathbf{T}_{WB}$ as $\mathbf{WB}$ for clarity.

7.1 Introduction

As we've motivated in Chapter 6, multi-robot co-localisation is essential for practically any multi-robot application where the robots interact with each other. Co-localisation accuracy, however, is heavily reliant on the quality of extrinsic calibration of the sensors (*e.g.* visual camera rigs, rangefinders) and the markers they can detect on other robots (*e.g.* AprilTags, reflective markers). While most works often take such extrinsic calibration for granted, in practice, the default in-factory calibration can only be precise to a certain degree. This is particularly important in multi-robot setups, where manual calibration becomes impractical and highly accurate in-factory; per-robot calibration incurs high operational costs.

Extending on the distributed localisation framework, Robot Web, which we've introduced in Chapter 6, in this chapter, we envision a system (see Figure 7.1) in which multiple robots co-localise themselves as they move and sense each other while simultaneously estimating and refining the extrinsic calibration of their sensor and their onboard marker on-the-fly. In summary, the contributions of our work are:

- A novel method for distributed multi-robot localisation and extrinsic calibration of both the sensor and the observed marker on the robots. Our approach builds on top of Robot Web, originally limited to pose variables in $\mathbf{SE}(2)$ given range-bearing observations. We extend the framework by simultaneously estimating the $\mathbf{SE}(3)$ pose of the robots and their extrinsic calibrations using Gaussian Belief Propagation (GBP) to further improve the accuracy of multi-robot localisation.

- We present a formulation of the inter-robot factor that avoids the sharing of the calibration variables amongst multiple robots, sparing communication effort between robots and thus enhancing the scalability of the system.

- We provide an extensive evaluation of our approach in comparison with other state-of-the-art alternatives and measure the performance of the method under extreme conditions such as a large number of communication failures, a large proportion of outlying measurements, and a limited communication range.

7.2 Related Works

We've reviewed multi-robot localisation in Section 6.2, hence in this section, we discuss multi-device calibration methods.

7.2.1 Multi-Device Calibration

Calibration is vital for robotic operations and, as such, the body of literature on the subject is vast. Due to space limitations, our literature review focuses on calibration processes that involve multiple cameras or multiple robots.

Accurate extrinsic calibration is often critical in multi-device systems. Different robots have different base frames, and within a robot, the exact position of the sensor and observable onboard marker may not be available. A common instance of this is hand-eye calibration. From a set of known relative transformations, the calibration process seeks to establish the undetermined relationship, often the relative transformation between the robot base frame and sensor frame [Shiu and Ahmad, 1989]. These methods can be extended to support multiple robots using iterative methods [Wang et al., 2014] or probabilistic approaches [Ma et al., 2018]. However, these methodologies primarily focus on offline settings where calibration precedes operational activities. In multi-robot setups, [Gowal et al., 2011] proposes a method to perform online calibration of infrared sensors while estimating the parameters of the underlying physical sensor model.

Multi-camera rigs are becoming increasingly popular as they can significantly extend the surrounding perceptive field for any robot and even directly yield stereo-depth capabilities provided there is view overlap. However, accurate calibration of such these rigs is often challenging, leading to several works on automatic offline calibration [Esquivel et al., 2007, Carrera et al., 2011, Lin et al., 2020]. The method in [Dang et al., 2009] carries out continuous self-calibration using an extended Kalman filter in a stereo setup. Beyond two cameras, self-calibration of multiple cameras extrinsic is achieved on an aerial vehicle in [Heng et al., 2015], whereas an information-theoretic approach described in [Dexheimer et al., 2022] is able to operate on a rig of eight cameras. Notably, while these methods are online, they predominantly address setups with a single robot with multiple onboard sensors, rather than a truly distributed, multi-robot system.

In the field of sensor networks, CaliBree [Miluzzo et al., 2008] performs fully distributed sensor calibration by measuring disagreement between uncalibrated and calibrated sensors upon rendezvous event. This method, however, is only limited to calibration and does not address the localisation of the devices. In [Devarajan et al., 2008], GBP is used in a distributed fashion for intrinsic calibration and refinement of the camera poses. The method solves structure-from-motion, where multiple cameras are stationary; hence, it is not applicable to online robotic applications with a moving onboard sensor. Non-parametric belief propagation is used in [Ihler et al., 2004] to perform calibration and localisation of sensors. The method is sampling-based; hence less efficient than GBP and assumes that the sensors are stationally. LaSLAT [Taylor et al., 2006] performs localisation and calibration of the sensors together with tracking of a target. In LaSLAT, the sensor poses are assumed to be static and are not suitable for localising multiple moving robots, which is the problem we address in this work.

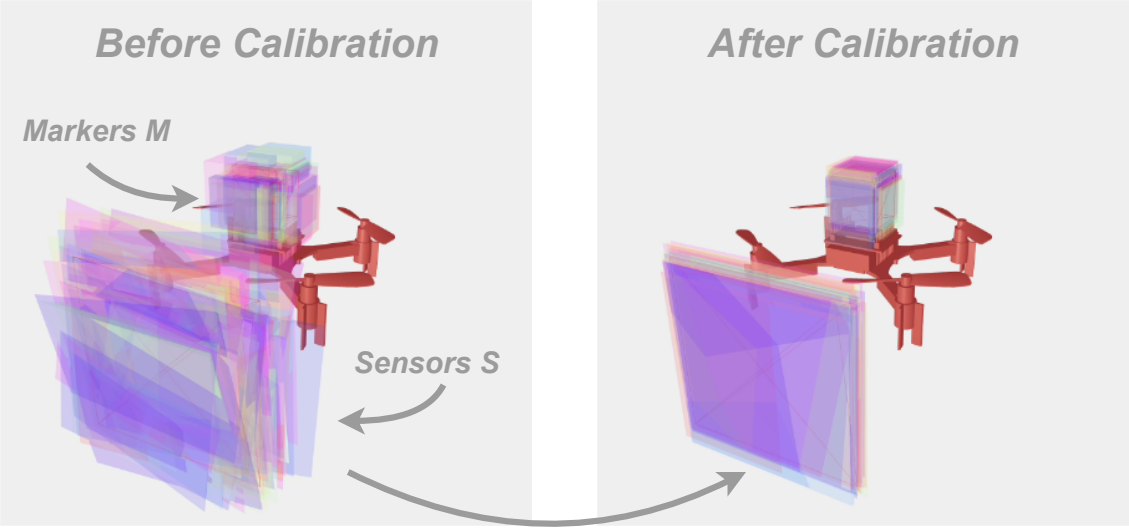


Figure 7.2: Example of calibration of the extrinsic of the sensors' pose and markers' position using the proposed method, where we artificially set the ground-truth extrinsics to be the same for visual clarity. We overlay the calibration estimates of 64 robots from randomly initialised states (left), and visualise the estimated extrinsics after the calibration (right).

7.3 Distributed Localistion and Extrinsic Calibration

7.3.1 Problem Formulation

This section details how multi-robot localisation and the extrinsic calibration of their onboard sensors and observable markers are simultaneously and distributedly performed. In the considered setting, each robot is equipped with a range-bearing sensor S that observes the other robot's onboard marker M. While the addressed setup is representative of realistic constraints of many robotic applications, here we describe the formulation in general terms, so that it can easily be extended to support more information-rich measurements such as direct relative transformations (*e.g.* using visual sensors and fiducial markers).

The relative transformation from the base of the robot B to its sensor S is $\mathbf{T}_{BS} \in \mathbf{SE}(3)$, where the notation $\mathbf{T}_{BS}$ represents the pose of S in the coordinate frame of B. Similarly, $\mathbf{t}_{BM} \in \mathbb{R}^3$ represents the marker position M relative to B. Since only range-bearing sensors are used in the current setup, the orientation of the marker is not observable and thus not included in this specific problem definition. When the sensor S^α in robot α observes the marker M^β in robot β, a relative measurement $\bar{\mathbf{z}}^{\alpha\beta}_{SM}$ is generated. The initial estimates of $\mathbf{T}_{BS}$ and $\mathbf{t}_{BM}$ are expected to be noisy due to inaccurate calibration. Our work optimises over the extrinsic calibration using the observations $\bar{\mathbf{z}}^{\alpha\beta}_{SM}$ robots accumulate over time as depicted in Figure 7.2.

Let $\Omega = \{\alpha, \beta, \gamma, \ldots\}$ be the set of robots, T be the number of considered time-steps, and $\mathbf{T}^{\omega}_{WBt}$ denote the pose of the base of the robot ω at time t in the world coordinates. To perform marginal inference over all, $\mathbf{T}^{\omega}_{BS}, \mathbf{t}^{\omega}_{BM}, \mathbf{T}^{\omega}_{WBt}, \forall\omega \in \Omega, \forall t \in \{1, \ldots T\}$, we consider the following factors.

Range Bearing Sensor

In our setup, robots can observe the other robots using range-bearing sensors. We use spherical coordinate (r, θ, ϕ), i.e. radial distance, azimuthal angle, and elevation angle respectively. All angles are parameterised using $\mathbf{SO}(2)$, hence; the range bearing measurement is $\bar{\mathbf{z}}^{\alpha\beta}_{SM} \in \langle \mathbb{R}, \mathbf{SO}(2), \mathbf{SO}(2) \rangle$, a composite manifold.

The range bearing factor relating the sensor $\mathbf{T}^{\alpha}_{WSt}$ and the marker $\mathbf{t}^{\beta}_{WMt}$ at time t is:

$$l_s(\mathbf{T}^{\alpha}_{WSt}, \mathbf{t}^{\beta}_{WMt}; \bar{\mathbf{z}}^{\alpha\beta}_{SM}) \propto \exp(-\frac{1}{2} \| \bar{\mathbf{z}}^{\alpha\beta}_{SM} \diamond h_s(\mathbf{T}^{\alpha}_{WSt}, \mathbf{t}^{\beta}_{WMt}) \|^2_{\Sigma_s}) \,, \tag{7.1}$$

where we use the notation $\diamond$ from [Solà et al., 2018], an operation on the composite manifold ($\ominus$ operation is applied to each block of composites separately).

Here, h_s is the function that predicts range bearing measurement between $\mathbf{T}^{\alpha}_{WSt}$ and $\mathbf{t}^{\beta}_{WMt}$. Let $\mathbf{t}^{\alpha}_{WSt} \in \mathbb{R}^3$, $\mathbf{R}^{\alpha}_{WSt} \in \mathbf{SO}(3)$ be translational, rotational the part of $\mathbf{T}^{\alpha}_{WSt}$ (and similarly for $\mathbf{T}^{\beta}_{WMt}$). The relative translation between $\mathbf{T}^{\alpha}_{WSt}, \mathbf{T}^{\beta}_{WMt}$ in coordinate frame of base of α is: $\mathbf{R}^{\alpha\top}_{WSt}(\mathbf{t}^{\beta}_{WMt} - \mathbf{t}^{\alpha}_{WSt}) = (\delta x, \delta y, \delta z)$. Hence, measurement prediction function is defined as: $h_s(\mathbf{T}^{\alpha}_{WB}, \mathbf{T}^{\beta}_{WB}) = (r, \theta, \phi)$, where $r = \|\mathbf{t}^{\beta}_{WB} - \mathbf{t}^{\alpha}_{WB}\|_2$, $\theta = \text{arctan2}(\delta y, \delta x)$, and $\phi = \frac{\pi}{2} - \arccos(\frac{\delta z}{r})$.

Robot Odometry

We assume that odometry measurements $\bar{\mathbf{T}}_{B_{t-1}B_t} \in \mathbf{SE}(3)$ (*e.g.* IMU/wheel odometry) are made available to each robot. An odometry factor penalises the deviation between observation $\bar{\mathbf{T}}_{B_{t-1}B_t}$ and the two estimated consecutive poses $\mathbf{T}_{WB_{t-1}}, \mathbf{T}_{WB_t} \in \mathbf{SE}(3)$:

$$l_o(\mathbf{T}_{WB_{t-1}}, \mathbf{T}_{WB_t}; \bar{\mathbf{T}}_{B_{t-1}B_t}) \propto \exp(-\frac{1}{2} \| \bar{\mathbf{T}}_{B_{t-1}B_t} \ominus (\mathbf{T}^{-1}_{WB_{t-1}} \mathbf{T}_{WB_t}) \|^2_{\Sigma_o}) \,. \tag{7.2}$$

This assumes that the odometry measurement is measured in the base frame B. This property can be enforced by choosing a suitable base frame given prior information about the wheel/IMU position. However, if the odometry is provided via the sensor S (i.e. visual odometry), we can replace the transformations in the base frame B with a transformation in the sensor from S.

Calibration Factor

In Equation 7.1, we have used $\mathbf{T}_{WSt}, \mathbf{t}_{WMt}$ position of sensor S and marker M in the world coordinate frame W at time t. A simple solution to obtain the position in the world coordinate is to use all $\mathbf{T}^{\alpha}_{WBt}, \mathbf{T}^{\alpha}_{BS}, \mathbf{T}^{\beta}_{WBt}, \mathbf{t}^{\beta}_{BM}$ inside the likelihood function, as shown in the left row of Figure 7.3. However, this has a clear disadvantage: robots must communicate both the calibration estimate and the pose estimate with each other. This not only doubles the inter-robot communication effort but also exposes internal states (i.e. sensor calibration) that do not need to be revealed to other robots. Furthermore, it creates small cycles which often leads to overconfidence [Weiss

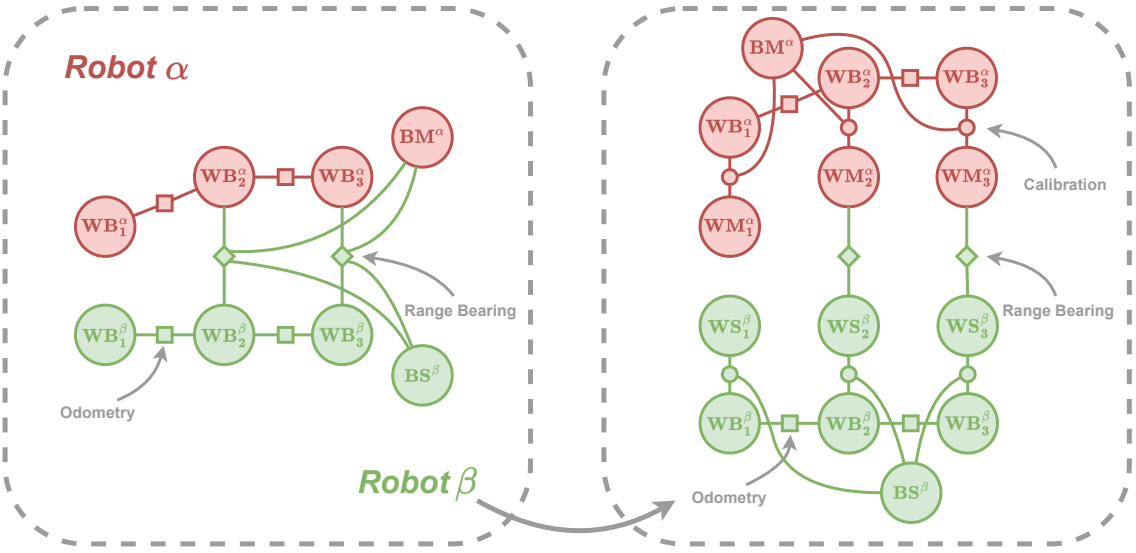


Figure 7.3: Reducing the inter-robot communication by restructuring the factor graph. We refer to $\mathbf{T}_{WB}$ as $\mathbf{WB}$ for clarity. **Left:** The inter-robot factor (range bearing) depends on four variables: the poses of the robots, the marker M, and the sensor S pose with respect to the robot base B. **Right** We introduce the marker and sensor variable in the world coordinate frame using Equation 7.3. We follow the same assumption as Chapter 6 that communication is the dominant cost we aim to reduce. Here, though the total number of variables increases, since the inter-robot factors depend on fewer variables, we reduce the communication requirements which makes our approach scalable.

and Freeman, 1999]. Hence, this motivates the redesign of a factor to only share the pose estimate of the sensor and the marker between robots.

The objective of the calibration is to find a transformation $\mathbf{T}_{BS} \in \mathbf{SE}(3)$ such that: $\mathbf{T}_{WS} = \mathbf{T}_{WB}\mathbf{T}_{BS}$. This relationship as a likelihood is defined as:

$$l_c(\mathbf{T}_{WS}, \mathbf{T}_{WB}, \mathbf{T}_{BS}) \propto \exp(-\frac{1}{2}\| \operatorname{Log}(\mathbf{T}_{WS}^{-1}\mathbf{T}_{WB}\mathbf{T}_{BS})\|^2_{\Sigma_c}) \,. \tag{7.3}$$

The likelihood for calibration of the marker can be derived in a similar way. This allows us to create a factor graph as illustrated in the right row of Figure 7.3, where only $\mathbf{T}_{WSt}, \mathbf{t}_{WMt}$ is connected between the robots. While this formulation increases the total number of variables in the factor graph, fewer variables are connected to the inter-robot factor, thus reducing the data transfer between the robots.

7.3.2 Adaptive Regulariser on the Factor

Due to the nature of $\mathbf{SE}(3)$, the objective function which we are minimising is non-linear and non-convex; challenging for any iterative optimisers, but especially for local ones such as GBP with no access to the global objective function. In our case, the Lie group extension of GBP Section 5.3 was insufficient to consistently reach convergence. Hence, here, we introduce an adaptive regularisation term in GBP to assist convergence. The idea presented is similar to diagonal loading [Johnson et al., 2009]; however, we adaptively change the priors we load for each factor.

Table 7.1: Accuracy of the proposed method ('Ours') and the global, centralised NLLS LM solver ('LM') at convergence as a function of the number of robots N and the enabling of autocalibration. Results include the RMSE ATE and ARE of the robot poses of their bases in the world frame $\mathbf{T}_{WB}$, the extrinsic calibration of their sensor $\mathbf{T}_{BS}$ and marker $\mathbf{t}_{BM}$ (only translation) where applicable.

N	$\mathbf{T}$	Initial		LM w/ Calib.		Ours w/ Calib.		Ours w/o Calib.	
		[m]	[deg]	[m]	[deg]	[m]	[deg]	[m]	[deg]
16	$\mathbf{T}_{WB}$	0.432	7.422	0.065	1.858	0.084	1.970	0.093	2.313
	$\mathbf{T}_{BS}$	0.080	8.852	0.023	1.156	0.027	1.268	–	–
	$\mathbf{t}_{BM}$	0.085	–	0.020	–	0.022	–	–	–
32	$\mathbf{T}_{WB}$	0.434	7.471	0.051	1.742	0.062	1.811	0.075	2.138
	$\mathbf{T}_{BS}$	0.082	8.856	0.021	1.035	0.025	1.251	–	–
	$\mathbf{t}_{BM}$	0.087	–	0.019	–	0.022	–	–	–
64	$\mathbf{T}_{WB}$	0.436	7.402	0.043	1.684	0.054	1.761	0.066	2.082
	$\mathbf{T}_{BS}$	0.083	8.810	0.020	0.969	0.025	1.214	–	–
	$\mathbf{t}_{BM}$	0.088	–	0.018	–	0.020	–	–	–
128	$\mathbf{T}_{WB}$	0.434	7.385	0.039	1.646	0.049	1.732	0.060	2.041
	$\mathbf{T}_{BS}$	0.085	8.740	0.018	0.969	0.022	1.202	–	–
	$\mathbf{t}_{BM}$	0.087	–	0.017	–	0.020	–	–	–

For each of the factors f_m, we add a zero-mean prior $\mathcal{N}^{-1}(0, \lambda_m \mathbf{I})$. The term λ_m^t is local to the factor and is updated adaptively based on the difference between the current local factor energy E_m^t and last iterations E_m^{t-1}:

$$\lambda_m^t = \begin{cases} \lambda_m^{t-1} \cdot \lambda_\uparrow & E_m^t - E_m^{t-1} > \epsilon_\lambda \\ \lambda_m^{t-1} / \lambda_\downarrow & \text{otherwise} \end{cases}, \tag{7.4}$$

where a threshold ϵ_λ is required to avoid the weighting from increasing when the factors' energy stops changing significantly near convergence, and $\lambda_\uparrow, \lambda_\downarrow$ are the increase, decrease factor respectively. Intuitively, at the beginning when far from optima, the adaptive regulariser encourages small descent steps. As the factors become more confident about their approximation of the curvature (i.e. made multiple successive descents), larger descent steps are performed. While the principle of this approach is the same as Levenberg-Marquardt, fundamental this weighting scheme is computed and applied purely locally, and the step is always taken even if the local energy increases. This way, no synchronisation or communication is required when applied distributedly.

Assuming that the objective function is strictly convex, the addition of the adaptive regularisation term will not change the optimal solution. As the GBP converges, $\lim_{\lambda_m \to 0} l_s(X_s; \bar{\mathbf{z}}_s) \mathcal{N}^{-1}(0, \lambda_m \mathbf{I}) = l_s(X_s; \bar{\mathbf{z}}_s)$, and $\lambda_m \to 0$ as the energy decreases or reaches local convergence.

7.4 Evaluation

We mainly evaluate our approach in a simulated environment with a vast number of robots, as obtaining the ground-truth extrinsic calibration and robot poses in the real-world for such experi-

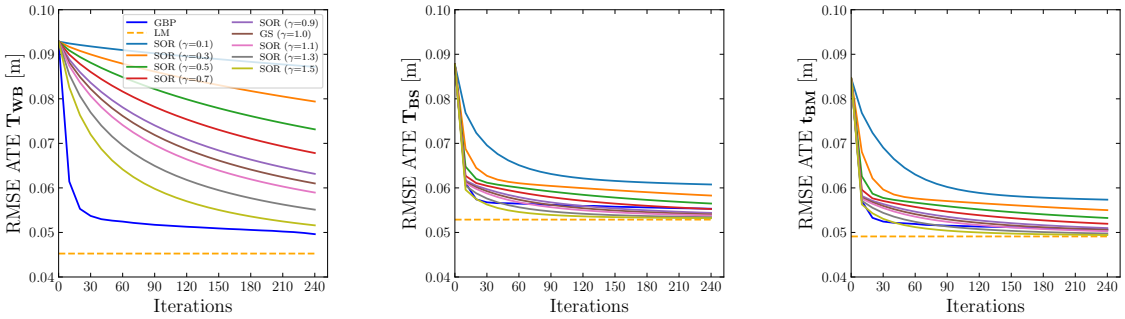


Figure 7.4: From left to right: RMSE ATE for $\mathbf{T}_{WB}$, $\mathbf{T}_{BS}$, $\mathbf{t}_{BM}$. RMSE ARE omitted as it follows the same trend. Comparison of different distributed alternatives (Final RMSE ATE of global, non-distributed LM shown for reference).

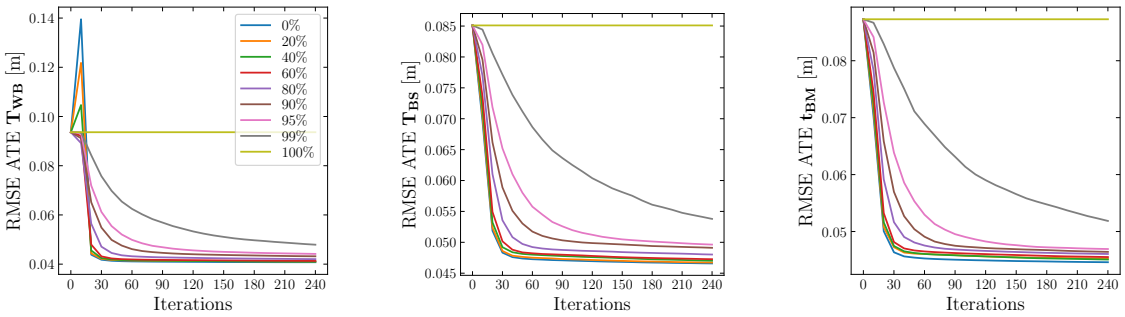


Figure 7.5: From left to right: RMSE ATE for $\mathbf{T}_{WB}$, $\mathbf{T}_{BS}$, $\mathbf{t}_{BM}$. RMSE ARE omitted as it follows the same trend. Analysis of robustness regarding communication failures by randomly dropping a percentage of the inter-robot GBP messages in each iteration. 100% indicates that all inter-robot messages are dropped, preventing co-localisation.

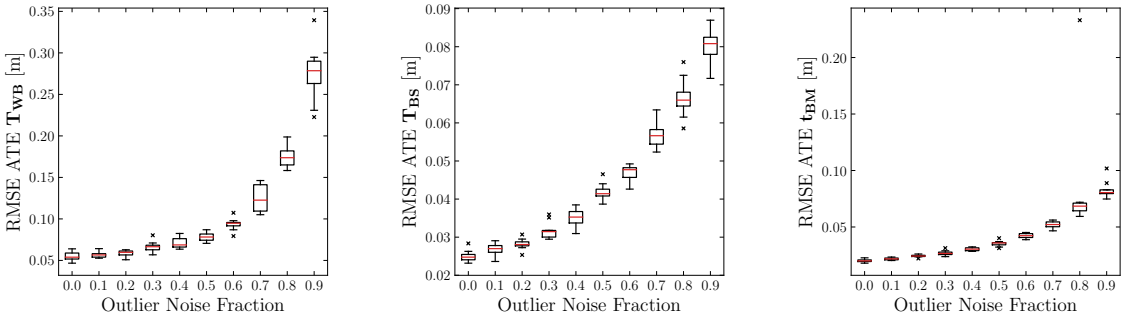


Figure 7.6: From left to right: RMSE ATE for $\mathbf{T}_{WB}$, $\mathbf{T}_{BS}$, $\mathbf{t}_{BM}$. RMSE ARE omitted as it follows the same trend. Effect of increasing the fraction of outlier noise. Non-Gaussian noise is added to the inter-robot sensor measurement to simulate outliers.

ments would be extremely challenging. As a verification of the applicability of our method to the real-world, we evaluate using UTIAS MR.CLAM dataset [Leung et al., 2011] .

To simulate sensor noise, observations are corrupted by applying zero-mean Gaussian noise. Odometry measurements are corrupted with noise with σ_B^t, a standard deviation of 0.01 meter per meter travelled for the translation, and with σ_B^R, a standard deviation of 1 degree per 90 degrees rotated for the rotation. Inter-robot measurements are corrupted in their range and bearing read-

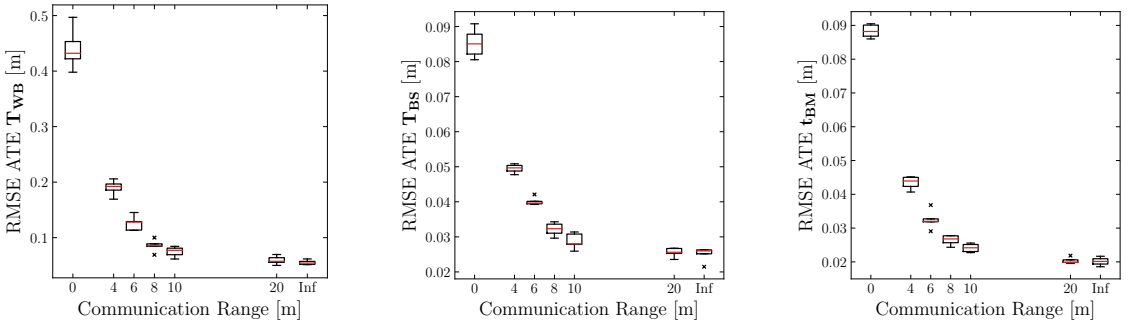


Figure 7.7: From left to right: RMSE ATE for $\mathbf{T}_{WB}$, $\mathbf{T}_{BS}$, $\mathbf{t}_{BM}$. RMSE ARE omitted as it follows the same trend. Impact on the overall accuracy when robots are limited to only communicating with peers within the specified range.

ings with a standard deviation of σ_s: (0.05m, 5°). 3D range bearing measurements (r, θ, ϕ) are restricted to the three closest observable robots, to imitate realistic and limited inter-robot observability. We further restrict the range-sensing to be limited to $|\theta| < 60°$ and $|\phi| < 60°$, to simulate the field of view limitations of, for instance, visual sensors. Robots are randomly initialised in translation and orientation within a $20m \times 20m \times 20m$ space. We assume that the initial pose of the robots is known to a certain degree, within a noisy initial guess with standard deviation 0.01m, 1° respectively for the translation and the rotation. We simulate the robots' motion by drawing random samples from a uniform distribution, $\mathcal{U}(0, 1)m$ for translational motion and $\mathcal{U}(-\pi, \pi)°$ for rotational motion, across all three dimensions. Finally, the initial calibration of sensor S and marker M also deviates from the ground truth with a standard deviation of the translational part of extrinsic of sensor σ_S^t and marker σ_M^t set to $0.05m$, and the standard deviation of the rotation part of the sensor frame $\sigma_S^R = 5°$.

To enhance the stability of GBP, for the adaptive regularisation, we use the default parameter of $\lambda_m = 10, \lambda_\downarrow = 9, \lambda_\uparrow = 11, \epsilon_\lambda = 10^{-4}$. While not sensitive to the choice of parameters, we found GBP to diverge in many cases without adaptive regularisation. Additionally, 30% of both internal and external GBP messages are randomly dropped, as an empirical heuristics to improve the convergence of the system [Ortiz et al., 2022]. Unless specified otherwise, for robustness against outlying measurements, we dynamically scale the information matrix of the range-bearing sensor factor using a DCS robust kernel [Agarwal et al., 2012] (Equation 6.3) with $\Phi = 10$. For each range-bearing measurement, the information matrix $\mathbf{\Lambda}_m$ is scaled by s_m^2.

In all the presented experiments, unless specified otherwise, we consider $N = 64$ robots that randomly execute 50 motions, incrementally growing the underlying factor graph (see Section 7.3.1) and performing 30 GBP message-passing iterations after each of these motions. The experimental results aggregate information from a total of 10 randomised runs, where we often report the average Root Mean Squared Error (RMSE) of Absolute Trajectory Error (ATE) and Absolute Rotation Error (ARE) to measure the accuracy of the system as described in [Choudhary et al., 2017] for multi-robot setup.

7.4.1 Comparison with Centralised Factor-Graph Solvers

Here we compare the proposed incremental GBP-based approach with a global Non-Linear Least Squares (NLLS) Levenberg-Marquardt (LM) solver (implemented in Theseus [Pineda et al., 2022]) that processes the full graph as a whole batch. We evaluate how the accuracy of the overall system varies as a function of the number of robots N and the effect resulting from enabling or disabling auto-calibration for the proposed method, i.e. whether the initial noisy calibration is optimisable or remains fixed, respectively. The robust kernel is disabled for this experiment to simplify the comparison. The accuracy of the different alternatives is compared in Table 7.1. Despite the proposed method being distributed and without any global, second-order perspective of the whole problem, experimental results show no significant differences with respect to the global LM solver at convergence. As expected, the larger the number of robots N, the higher the accuracy of all methods as the underlying factor graph becomes denser and thus, more information-rich. Observe that the proposed GBP-based approach is still able to profit from a denser graph despite including more cycles. We additionally report the results of our method while considering that the noisy initial calibration is correct. This yields obviously worse results than when we optimise the graph which considers the calibration parameters.

7.4.2 Comparison with Distributed Factor-Graph Solvers

We compare our method against other distributed solvers: block Gauss-Seidel (GS) and its relaxations block Successive Over-Relaxation (SOR) [Bertsekas and Tsitsiklis, 2015]. In GBP messages are exchanged in parallel and thus do not require coordinated updates. We favour GS and SOR by counting each iteration as an ordered sweep, where robots sequentially exchange their updated state in a specific order. In this comparison, all the methods are provided with the whole graph to be optimised from the beginning instead of incrementally growing and solving the problem, with 16 robots making 10 random motions. We use the same relaxation parameter as reported in [Choudhary et al., 2017]. The robust kernel is disabled for this experiment to simplify the comparison.

Results are presented in the Figure 7.4. GBP shows a faster convergence rate than GS and SOR in the number of iterations (the aforementioned global and centralised LM method is also shown for reference). Presented results match prior comparisons between GBP and Successive Over-Relaxation in [Weiss and Freeman, 1999]. Note that GS and SOR produce marginally better extrinsic at by trading off a significantly worse body frame localisation.

7.4.3 Robustness Analysis

Communication Failure and Asynchronicity

We analyse the robustness of the system regarding potential communication failures, modelled by randomly dropping a percentage of the inter-robot GBP messages in each iteration, and present the results in the Figure 7.5. In this experiment, the whole graph is available from the beginning of the proposed algorithm with 64 robots and 10 random motions to clearly identify the convergence trends. The behaviour of the system remains largely unaffected by communication failure up to around 80%, communication failure. ATE of $\mathbf{T}_{WB}$ initially increases as poses are initially uncertain and are down-weighted by the robust kernel. However, within a few iterations, the poses are correctly optimised, reducing the ATE. Even at an extremely high communication failure rate, the RMSE ATE still gradually decreases as we perform more iterations and thus, more rounds of communication. The experiment further demonstrates the asynchronicity of our approach, where the message order does not significantly impact the overall performance. This is a crucial property required for real-world deployment, where the communication channel is potentially unreliable, especially at scale.

Robustness to Outlier Measurements

As real-world inter-robot sensing is often challenging (*e.g.* misidentification, sensor failure), we investigate the robustness of the system to extreme, non-Gaussian outlier measurements following a uniform noise. As presented in the Figure 7.6, results indicate that, while performance is reasonably impacted as the fraction of outliers increases, the system remains stable even for an extremely high percentage of non-Gaussian outliers. Even at 40%, we observe a relatively small increase in the error compared to no outlying noise, demonstrating the robustness of our approach.

Communication Range

To mimic realistic, real-world conditions, we further limit the communication radius of the robots to 4, 6, 8, 10 and 20m and report the results on the Figure 7.7, including also no communication (0m) and infinite communication range ('Inf') for completeness. While a longer communication range proves to be indeed beneficial, our method is able to optimise all the parameters effectively even with a severely limited communication radius. For reference, only 24% of the robots are within communication range at a 10m radius whereas the percentage increases to 85% at a 20m radius and yet such a drastic increase only yields insignificant returns in terms of accuracy.

The imposed limit on the communication range also tests the system's asynchronicity. Since an agent cannot communicate at the time of observation if the other agent is too far away, the agent needs to wait for a rendezvous event in order to exchange information. Our result hence further highlights that our system is capable of handling asynchronous events.

Table 7.2: Evaluation of our method on real-world data. Results include the RMSE ATE of our system with and without the autocalibration enabled, on UTIAS MR.CLAM dataset 1-4. The noise column indicates whether a noise was artificially added to the sensor calibration to simulate an uncalibrated system.

Noise	Auto Calib.	1	2	3	4
✗	✗	**0.102**	0.0976	**0.0690**	0.0712
✗	✓	**0.102**	**0.0967**	0.0706	**0.0694**
✓	✗	0.122	0.121	0.0974	0.0855
✓	✓	**0.111**	**0.120**	**0.0825**	**0.0809**

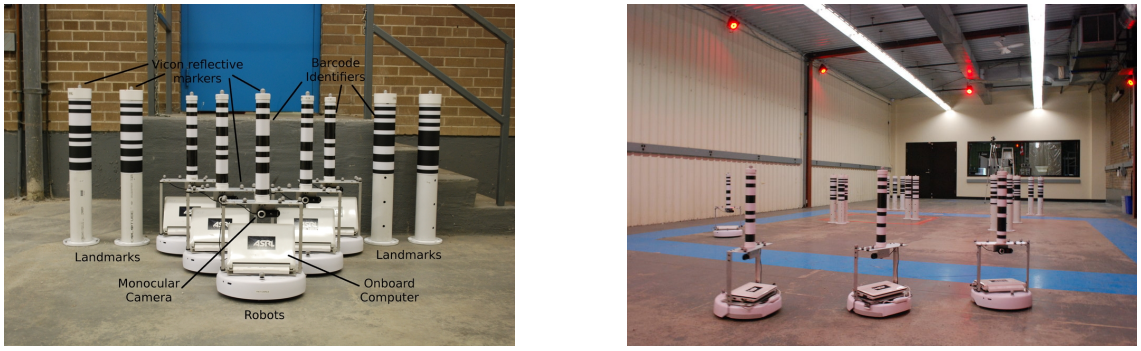


Figure 7.8: Robots used in the UTIAS MR.CLAM dataset. Each robot is equipped with a monocular camera and a barcode identifier. Images from [Leung et al., 2011].

Real-world Experiments

To verify the applicability of our approach to real-world data, we have performed localisation and auto-calibration using UTIAS MR.CLAM dataset [Leung et al., 2011]. As the robots are ground vehicles (Figure 7.8), we model them with $\mathbf{SE}(2)$ poses and 2D range-bearing observations following Section 6.5.1. We use datasets 1-4, where the landmarks and the robots are randomly scattered. We set $\sigma_B^t = (0.05\text{m}, 0.01\text{m}), \sigma_B^R = 5°, \sigma_s = (0.08\text{m}, 2°)$. The dataset is subsampled at 1s intervals, and we use sliding window-based GBP with a window size of 30.

As presented in Table 7.2, as the robots are calibrated, autocalibration does not yield better RMSE ATE. We simulate an uncalibrated system by artificially adding noise to the sensor calibration with a standard deviation of 0.05m and $10°$ for translation and rotation respectively. While this manipulation of calibration is artificial, this data still contains challenging real-world sensor and odometry noise. In such a case, auto-calibration reduces the ATE, indicating that our method has successfully filtered out the biases even with real-sensor data.

7.5 Conclusion

In this chapter, we presented a method for online, simultaneous localisation and automatic extrinsic calibration of sensors and observable markers, by building on our previous work Robot Web from Chapter 6. Our work performs distributed and asynchronous inference on the factor

graph using GBP, and we have demonstrated its robustness against large amounts of communication failure, outlying measurements, and restricted communication ranges. One of the limitations of the autocalibration is the observability problem. For example, if two robots continuously move in a straight line, we will not be able to isolate the extrinsic calibration of the sensor/marker from the body pose. In a multi-device setting, since more devices are moving in different directions, these problems are less likely to occur; however, automatically identifying such an ambiguous state would be an interesting future direction.

Distributed and asynchronous properties of GBP offer attractive features for multi-robot systems. Automatic calibration ensures that the robots require as little maintenance as possible and the accurate localisation provides the basis required for multi-robot interaction.

CHAPTER 8

Conclusion and Future Directions

In this thesis, we explored *Distributed Spatial AI*. To conclude, we will revisit each of the contributions and discuss the future directions of research.

We started by investigating near-sensor processing in Chapter 3, where we bypass the expensive analog-to-digital conversion of the pixel intensities by performing analog computation directly on the focal-plane. Only the meaningful visual features are transferred from the focal-plane to the host device to save energy. Our work demonstrated that even though only a few bits of information are transferred from the camera, Visual Odometry (VO) pipeline can still be made robust by running the entire system at 300 FPS. In Chapter 4, we push the boundary of what the fidelity a Simultaneous Localisation and Mapping (SLAM) system can capture by demonstrating the first application of 3D Gaussians Splatting (3DGS) for monocular/RGB-D SLAM. Our work operates live at 3 FPS, and the continuous stream of image data from a moving camera is processed and compressed into a single 3D representation. While the compression is lossy and the re-rendered images are not perfect, they are photorealistic, and we can run an off-the-shelf pre-trained foundation models such as Segment Anything [Kirillov et al., 2023] on the rendered image.

Both Chapter 3 and Chapter 4 approached the problem of data compression, however, from different ends. BIT-VO aggressively compresses the image data, even before they are digitised, to minimise energy consumption and data transfer. On the other hand, Gaussian Splatting SLAM assumes the availability of all the pixel intensities and investigates how we can incrementally combine these data into a single coherent 3D representation while simultaneously tracking against them.

To address the problem of scalability in Distributed Spatial AI, in Chapter 5, we introduce Gaussian Belief Propagation (GBP), a message-passing-based algorithm, as a promising candidate for Distributed Spatial AI systems. Extending GBP to support the Lie groups, in Chapter 6, we present *Robot Web*, a framework for many device localisation. We demonstrate how 1000s of robots can be localised using GBP, even under challenging situations, such as communication failures and large amounts of outlying measurements. We demonstrate the applicability of our framework on real-world robots, where autonomous robots localise each other to stay on a square trajectory. We extend Robot Web in Chapter 7 to perform autocalibration of the sensors' and markers' extrinsic

while simultaneously localising, demonstrating that our Lie group formulation does indeed work with $\mathbf{SE}(3)$. We focused on scalability and simplicity in the Chapter 6 and Chapter 7. Taking inspiration from the World Wide Web, we envision a system where each device broadcasts selected information, which can be asynchronously received and integrated by neighbouring devices. GBP is an asynchronous algorithm and can converge even if the message-passing schedule is random. The sender of the messages does not need to be acknowledged; hence, GBP can operate using a broadcast-only communication model. Furthermore, individual nodes only communicate with the adjacent neighbours; thus, the devices operate and co-localise even without knowing how many devices are participating. These properties allow Robot Web to be scalable, robust, and applicable to real-world.

Looking forward, there are many interesting research directions. Bringing the ideas of near-sensor computation and GBP together, we can design a general-purpose vision sensor capable of performing probabilistic inference on the focal-plane. Since GBP performs better if the graph diameter is small [Scona et al., 2022], we imagine a sensor-processor with neighbouring connectivity and a couple of long-range, random connectivities. As long as the graph is connected, the 'small world' property of a graph allows messages to be delivered from any pixel to any pixel in a few hops, supporting dynamically changing topology of the underlying problem, similar to the idea of routing tile in [Ortiz et al., 2022]. Such sensors can perform challenging inference tasks such as visual odometry or even run CNN inference and training using a deep factor graph trained via GBP [Nabarro et al., 2023].

In Gaussian Splatting SLAM, the scene representation is explicit 3D Gaussians, meaning these Gaussians can be distributed amongst many devices. Using distributed optimisation to refine the 3D Gaussians, we can create a distributed dense SLAM with photorealistic rendering. This approach has the potential to achieve a level of fidelity that far surpasses what is currently possible with the state-of-the-art in multi-robot SLAM. High fidelity is crucial for applications such as inspection, and since the reconstruction is photorealistic, we can run off-the-shelf pre-trained models to perform tasks such as segmentation and anomaly detection on the novel-view synthesised image.

The recent advancements in AR/VR and robotics is making the necessity of local computation and distributed algorithm ever more obvious. With so many devices requiring spatial intelligence, distributed approaches like Robot Web is essential, both from scalability and privacy perspective. Currently, we focused only on distributed localisaton, however; extending the approach into full SLAM system, or even further into a dense 3D reconstruction by replacing gradient descent of Gaussian Splatting with GBP will be and interesting future direction.

Bibliography

[Absil et al., 2008] Absil, P.-A., Mahony, R., and Sepulchre, R. (2008). *Optimization algorithms on matrix manifolds*. Princeton University Press. 84

[Agarwal et al., 2012] Agarwal, P., Tipaldi, G. D., Spinello, L., Stachniss, C., and Burgard, W. (2012). Robust map optimization using dynamic covariance scaling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 101, 107, 114, 128

[Agarwal et al., 2010] Agarwal, S., Mierle, K., and Others (2010). *Ceres Solver*. http://ceres-solver.org [Accessed: 2024-01-28]. 51, 84, 98

[Amestoy et al., 1996] Amestoy, P. R., Davis, T. A., and Duff, I. S. (1996). An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905. 29

[Andersson and Nygards, 2008] Andersson, L. A. A. and Nygards, J. (2008). C-SAM : Multi-robot SLAM using square root information smoothing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 94

[Aragues et al., 2011] Aragues, R., Carlone, L., Calafiore, G., and Sagues, C. (2011). Multi-agent localization from noisy relative pose measurements. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 9

[Bae and Davison, 2024] Bae, G. and Davison, A. J. (2024). Rethinking inductive biases for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 16, 17

[Bailey et al., 2011] Bailey, T., Bryson, M., Mu, H., Vial, J., McCalman, L., and Durrant-Whyte, H. (2011). Decentralised cooperative localisation for heterogeneous teams of mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 94

[Bänninger et al., 2023] Bänninger, P., Alzugaray, I., Karrer, M., and Chli, M. (2023). Cross-agent relocalization for decentralized collaborative SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 9, 95

[Barfoot, 2017] Barfoot, T. D. (2017). *State Estimation for Robotics*. Cambridge University Press. 39, 84, 85, 96

[Barooah and Hespanha, 2005] Barooah, P. and Hespanha, J. (2005). Distributed estimation from relative measurements in sensor networks. In *International Conference on Intelligent Sensing and Information Processing*. 9

[Berrou et al., 1993] Berrou, C., Glavieux, A., and Thitimajshima, P. (1993). Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *IEEE International Conference on Communications*, volume 2, pages 1064–1070 vol.2. 7

[Bertsekas and Tsitsiklis, 2015] Bertsekas, D. and Tsitsiklis, J. (2015). *Parallel and distributed computation: numerical methods*. Athena Scientific. 9, 129

[Bertsekas, 1997] Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334. 27

[Bickson, 2008] Bickson, D. (2008). *Gaussian belief propagation: Theory and Application*. PhD thesis, PhD thesis, The Hebrew University of Jerusalem. 9

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc. 30, 33, 34

[Bose et al., 2017] Bose, L., Chen, J., Carey, S. J., Dudek, P., and Mayol-Cuevas, W. (2017). Visual Odometry for Pixel Processor Arrays. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 4614–4622. 14, 47, 54

[Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122. 9

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 56

[Caceres et al., 2011] Caceres, M. A., Penna, F., Wymeersch, H., and Garello, R. (2011). Hybrid cooperative positioning based on distributed belief propagation. *IEEE Journal on Selected Areas in Communications*, 29(10):1948–1958. 95

[Calafiore et al., 2010] Calafiore, G. C., Carlone, L., and Wei, M. (2010). A distributed gradient method for localization of formations using relative range measurements. In *IEEE International Symposium on Computer-Aided Control System Design*. 8

[Calafiore et al., 2012] Calafiore, G. C., Carlone, L., and Wei, M. (2012). A distributed technique for localization of agent formations from relative range measurements. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 42(5):1065–1076. 8

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 778–792. Springer. 49, 56

[Carey et al., 2013a] Carey, S. J., Barr, D. R., Wang, B., Lopich, A., and Dudek, P. (2013a). Mixed signal SIMD processor array vision chip for real-time image processing. *Analog Integrated Circuits and Signal Processing*, 77(3):385–399. 14

[Carey et al., 2013b] Carey, S. J., Lopich, A., Barr, D. R., Wang, B., and Dudek, P. (2013b). A 100000 fps vision sensor with embedded 535GOPS/W 256×256 SIMD processor array. In *IEEE Symposium on VLSI Circuits (VLSIC)*, pages 182–183. 14, 46, 52

[Carlone et al., 2015a] Carlone, L., Rosen, D. M., Calafiore, G., Leonard, J. J., and Dellaert, F. (2015a). Lagrangian duality in 3d slam: Verification techniques and optimal solutions. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. IEEE. 110

[Carlone et al., 2015b] Carlone, L., Tron, R., Daniilidis, K., and Dellaert, F. (2015b). Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 9, 111, 112

[Carrera et al., 2011] Carrera, G., Angeli, A., and Davison, A. J. (2011). SLAM-based automatic extrinsic calibration of a multi-camera rig. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 122

[Castillo-Elizalde et al., 2021] Castillo-Elizalde, H., Liu, Y., Bose, L., and Mayol-Cuevas, W. (2021). Weighted Node Mapping and Localisation on a Pixel Processor Array. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 14

[Censi and Scaramuzza, 2014] Censi, A. and Scaramuzza, D. (2014). Low-latency event-based visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 11

[Ćesić et al., 2017] Ćesić, J., Marković, I., Bukal, M., and Petrović, I. (2017). Extended information filter on matrix Lie groups. *Automatica*, 82:226–234. 96

[Chen et al., 2017] Chen, J., Carey, S., and Dudek, P. (2017). Feature extraction using a portable vision system. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 14, 47, 49

[Choudhary et al., 2015] Choudhary, S., Carlone, L., Christensen, H. I., and Dellaert, F. (2015). Exactly sparse memory efficient slam using the multi-block alternating direction method of multipliers. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. IEEE. 9

[Choudhary et al., 2017] Choudhary, S., Carlone, L., Nieto, C., Rogers, J., Christensen, H., and Dellaert, F. (2017). Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *International Journal of Robotics Research (IJRR)*, 36(12):1286–1311. 9, 94, 110, 111, 112, 128, 129

[Cieslewski et al., 2018] Cieslewski, T., Choudhary, S., and Scaramuzza, D. (2018). Data-efficient decentralized visual SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 9, 94

[Cunningham et al., 2013] Cunningham, A., Indelman, V., and Dellaert, F. (2013). DDF-SAM 2.0: Consistent distributed smoothing and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 10, 95

[Cunningham et al., 2010] Cunningham, A., Paluri, M., and Dellaert, F. (2010). DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 10, 95

[Czarnowski et al., 2020] Czarnowski, J., Laidlow, T., Clark, R., and Davison, A. J. (2020). Deepfactors: Real-time probabilistic dense monocular SLAM. *IEEE Robotics and Automation Letters*, 5(2):721–728. 64

[Dai et al., 2017] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017). Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(4):1. 11, 15

[Dang et al., 2009] Dang, T., Hoffmann, C., and Stiller, C. (2009). Continuous stereo self-calibration by camera parameter tracking. *IEEE Transactions on Image Processing*, 18(7):1536–1550. 122

[Davis et al., 2004] Davis, T. A., Gilbert, J. R., Larimore, S. I., and Ng, E. G. (2004). Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30:377–380. 29

[Davison, 2018] Davison, A. J. (2018). Futuremapping: The computational structure of spatial ai systems. *arXiv preprint arXiv:1803.11288*. 1

[Davison et al., 2007] Davison, A. J., Molton, N. D., Reid, I., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067. 10, 46

[Davison and Ortiz, 2019] Davison, A. J. and Ortiz, J. (2019). FutureMapping 2: Gaussian Belief Propagation for Spatial AI. *arXiv preprint arXiv:1910.14139*. 77, 80, 90, 98, 104, 107

[Debrunner et al., 2019] Debrunner, T., Saeedi, S., Bose, L., Davison, A. J., and Kelly, P. H. J. (2019). Camera Tracking on Focal-Plane Sensor-Processor Arrays. In *In Proceedings of the Workshop on Programmability and Architectures for Heterogeneous Multicores (MULTIPROG)*. 14, 47

[Deisenroth et al., 2020] Deisenroth, M. P., Faisal, A. A., and Ong, C. S. (2020). *Mathematics for Machine Learning*. Cambridge University Press. 5, 6, 24, 25

[Dellaert, 2005] Dellaert, F. (2005). Square Root SAM. In *Proceedings of Robotics: Science and Systems (RSS)*. 26

[Dellaert, 2012] Dellaert, F. (2012). Factor graphs and GTSAM, a hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology. 9, 84, 98

[Dellaert, 2021] Dellaert, F. (2021). Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):141–166. 94, 103

[Dellaert and Kaess, 2017] Dellaert, F. and Kaess, M. (2017). Factor Graphs for Robot Perception. *Foundations and Trends in Robotics*, 6(1–2):1–139. 5, 6, 94

[Devarajan et al., 2008] Devarajan, D., Cheng, Z., and Radke, R. J. (2008). Calibrating distributed camera networks. *Proceedings of the IEEE*, 96(10):1625–1639. 122

[Dexheimer and Davison, 2023] Dexheimer, E. and Davison, A. J. (2023). Learning a Depth Covariance Function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 70, 71

[Dexheimer et al., 2022] Dexheimer, E., Peluse, P., Chen, J., Pritts, J., and Kaess, M. (2022). Information-theoretic online multi-camera extrinsic calibration. *IEEE Robotics and Automation Letters*, 7(2):4757–4764. 122

[Dominguez-Castro et al., 1997] Dominguez-Castro, R., Espejo, S., Rodriguez-Vazquez, A., Carmona, R. A., Foldesy, P., Zarándy, Á., Szolgay, P., Szirányi, T., and Roska, T. (1997). A 0.8-/spl mu/m CMOS two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage. *IEEE Journal of Solid-State Circuits (JSSC)*, 32(7):1013–1026. 46

[Dudek and Hicks, 2000] Dudek, P. and Hicks, P. J. (2000). A cmos general-purpose sampled-data analog processing element. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5):467–473. 46

[Dudek and Hicks, 2005] Dudek, P. and Hicks, P. J. (2005). A general-purpose processor-per-pixel analog SIMD vision chip. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(1):13–20. 14

[El-Desouki et al., 2009] El-Desouki, M., Deen, M., Fang, Q., Liu, L., Tse, F., and Armstrong, D. (2009). CMOS image sensors for high speed applications. *Sensors*, 9:430–44. 14

[Engel et al., 2017] Engel, J., Koltun, V., and Cremers, D. (2017). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 15, 64, 67, 70, 71

[Eriksson et al., 2016] Eriksson, A., Bastian, J., Chin, T.-J., and Isaksson, M. (2016). A consensus-based framework for distributed bundle adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1754–1762. 9

[Esquivel et al., 2007] Esquivel, S., Woelk, F., and Koch, R. (2007). Calibration of a Multi-camera Rig from Non-overlapping Views. In *Pattern Recognition. DAGM*, pages 82–91. 122

[Eustice et al., 2005] Eustice, R. M., Singh, H., and Leonard, J. J. (2005). Exactly Sparse Delayed State Filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 25

[Fan and Murphey, 2020] Fan, T. and Murphey, T. (2020). Majorization minimization methods for distributed pose graph optimization with convergence guarantees. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 8, 111

[Fan et al., 2023] Fan, T., Ortiz, J., Hsiao, M., Monge, M., Dong, J., Murphey, T., and Mukadam, M. (2023). Decentralization and acceleration enables large-scale bundle adjustment. In *Proceedings of Robotics: Science and Systems (RSS)*. 8

[Felzenszwalb and Huttenlocher, 2006] Felzenszwalb, P. F. and Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International Journal of Computer Vision (IJCV)*, 70:41–54. 7

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. 51

[Forster et al., 2014] Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. 15, 52, 64

[Forster et al., 2016] Forster, C., Zhang, Z., Gassner, M., Werlberger, M., and Scaramuzza, D. (2016). SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems. *IEEE Transactions on Robotics (T-RO)*, 33(2):249–265. 52

[Fridovich-Keil et al., 2022] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. (2022). Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 16

[Gallego et al., 2020] Gallego, G., Delbruck, T., Orchard, G. M., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A., Conradt, J., Daniilidis, K., and Scaramuzza, D. (2020). Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1–26. 11, 45

[Gallego et al., 2017] Gallego, G., Lund, J. E., Mueggler, E., Rebecq, H., Delbruck, T., and Scaramuzza, D. (2017). Event-based, 6-dof camera tracking from photometric depth maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(10):2402–2412. 11

[Gavin, 2019] Gavin, H. P. (2019). The levenberg-marquardt algorithm for nonlinear least squares curve-fitting problems. *Department of civil and environmental engineering, Duke University*, 19. 29

[Gemeiner et al., 2008] Gemeiner, P., Davison, A. J., and Vincze, M. (2008). Improving Localization Robustness in Monocular SLAM Using a High-Speed Camera. In *Proceedings of Robotics: Science and Systems (RSS)*. 44

[Goodfellow et al., 2016] Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org. 3

[Gowal et al., 2011] Gowal, S., Prorok, A., and Martinoli, A. (2011). Two-phase online calibration for infrared-based inter-robot positioning modules. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 122

[Greatwood et al., 2017] Greatwood, C., Bose, L., Richardson, T., Mayol-Cuevas, W., Chen, J., Carey, S. J., and Dudek, P. (2017). Tracking control of a UAV with a parallel visual processor. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 4248–4254. 14

[Greatwood et al., 2018] Greatwood, C., Bose, L., Richardson, T., Mayol-Cuevas, W., Chen, J., Carey, S. J., and Dudek, P. (2018). Perspective Correcting Visual Odometry for Agile MAVs using a Pixel Processor Array. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 987–994. 47

[Grisetti et al., 2007] Grisetti, G., Stachniss, C., Grzonka, S., Burgard, W., et al. (2007). A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*. 8

[Grupp, 2017] Grupp, M. (2017). evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo. 53

[Handa et al., 2012] Handa, A., Newcombe, R. A., Angeli, A., and Davison, A. J. (2012). Real-time camera tracking: When is high frame-rate best? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 222–235. Springer. 3, 44

[Heng et al., 2015] Heng, L., Lee, G. H., and Pollefeys, M. (2015). Self-calibration and visual SLAM with a multi-camera system on a micro aerial vehicle. *Autonomous Robots*, 39:259–277. 122

[Hooker, 2020] Hooker, S. (2020). The Hardware Lottery. In *arXiv:1911.05248*. 12

[Hsueh et al., 2022] Hsueh, H.-Y., Toma, A.-I., Ali Jaafar, H., Stow, E., Murai, R., Kelly, P. H., and Saeedi, S. (2022). Systematic comparison of path planning algorithms using pathbench. *Autonomous Robots*, 36(11):566–581. 18

[Huang et al., 2021] Huang, J., Huang, S.-S., Song, H., and Hu, S.-M. (2021). Di-fusion: Online implicit 3d reconstruction with deep priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 70, 71

[Huang et al., 2022] Huang, T., Gu, H., and Nelson, B. (2022). Increasingly intelligent micromachines. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:25.1–25.32. 118

[Hughes et al., 2022] Hughes, N., Chang, Y., and Carlone, L. (2022). Hydra: A real-time spatial perception system for 3d scene graph construction and optimization. *Proceedings of Robotics: Science and Systems (RSS)*. 11

[Ihler et al., 2004] Ihler, A., Fisher, J., Moses, R., and Willsky, A. (2004). Nonparametric belief propagation for self-calibration in sensor networks. In *Third International Symposium on Information Processing in Sensor Networks*, pages 225–233. 122

[Indelman et al., 2014] Indelman, V., Nelson, E., Michael, N., and Dellaert, F. (2014). Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 94

[Jaynes, 2003] Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press. 23

[Johari et al., 2023] Johari, M. M., Carta, C., and Fleuret, F. (2023). ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 64, 70, 71, 72

[Johnson et al., 2009] Johnson, J. K., Bickson, D., and Dolev, D. (2009). Fixing convergence of gaussian belief propagation. In *2009 IEEE International Symposium on Information Theory*, pages 1674–1678. IEEE. 125

[Kaess et al., 2012] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2012). iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *International Journal of Robotics Research (IJRR)*, 31(2):216–235. 94

[Kaspar et al., 2021] Kaspar, C., Ravoo, B. J., van der Wiel, W. G., Wegner, S. V., and Pernice, W. H. P. (2021). The rise of intelligent matter. *Nature*, 594:345–355. 118

[Keller et al., 2013] Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., and Kolb, A. (2013). Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proceedings of the International Conference on 3D Vision (3DV)*. 15, 63, 64

[Kerbl et al., 2023] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–14. 63, 70

[Keselman and Hebert, 2022] Keselman, L. and Hebert, M. (2022). Approximate differentiable rendering with algebraic surfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 16

[Kim et al., 2010] Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S. (2010). Multiple relative pose graphs for robust cooperative mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 94

[Kim et al., 2014] Kim, H., Handa, A., Benosman, R., Ieng, S.-H., and Davison, A. (2014). Simultaneous mosaicing and tracking with an event camera. In *Proceedings of the British Machine Vision Conference (BMVC)*. 11

[Kim et al., 2016] Kim, H., Leutenegger, S., and Davison, A. J. (2016). Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 349–364. 11

[Kim et al., 2021] Kim, Y. M., Bahn, S., and Yun, M. H. (2021). Wearing comfort and perceived heaviness of smart glasses. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 31(5):484–495. 2

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 27, 66

[Kirillov et al., 2023] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. (2023). Segment anything. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 16, 133

[Kitamura et al., 2012] Kitamura, K., Watabe, T., Sawamoto, T., Kosugi, T., Akahori, T., Iida, T., Isobe, K., Watanabe, T., Shimamoto, H., Ohtake, H., Aoyama, S., Kawahito, S., and Egami, N. (2012). A 33-Megapixel 120-Frames-Per-Second 2.5-Watt CMOS Image Sensor With Column-Parallel Two-Stage Cyclic Analog-to-Digital Converters. *IEEE Transactions on Electron Devices*, 59(12):3426–3433. 46

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 10, 11, 51, 52

[Knuth and Barooah, 2013] Knuth, J. and Barooah, P. (2013). Collaborative localization with heterogeneous inter-robot measurements by riemannian optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 8

[Kümmerle et al., 2011] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g^2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 5, 84, 98

[Lajoie et al., 2020] Lajoie, P., Ramtoula, B., Chang, Y., Carlone, L., and Beltrame, G. (2020). DOOR-SLAM: Distributed online and outlier resilient SLAM for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663. 9, 94

[Lajoie et al., 2022] Lajoie, P.-Y., Ramtoula, B., Wu, F., and Beltrame, G. (2022). Towards collaborative simultaneous localization and mapping: a survey of the current research landscape. *ArXiv*, abs/2108.08325. 94

[Lazaro et al., 2013] Lazaro, M. T., Paz, L. M., Pinies, P., Castellanos, J. A., and Grisetti, G. (2013). Multi-robot SLAM using condensed measurements. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 10

[Leung et al., 2011] Leung, K. Y., Halpern, Y., Barfoot, T. D., and Liu, H. H. (2011). The UTIAS multi-robot cooperative localization and mapping dataset. *International Journal of Robotics Research (IJRR)*, 30(8):969–974. 127, 131

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). BRISK: Binary Robust Invariant Scalable Keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2548–2555. 49, 56

[Levenberg, 1944] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168. 29

[Li et al., 2014] Li, B., Wu, N., Wang, H., Kuang, J., and Xing, C. (2014). Gaussian message-based cooperative localization on factor graph in wireless sensor networks. In *International Conference on Wireless Communications and Signal Processing*. 95

[Li et al., 2015] Li, B., Wu, N., Wang, H., Tseng, P.-H., and Kuang, J. (2015). Gaussian message passing-based cooperative localization on factor graph in wireless networks. *Signal Processing*, 111:1–12. 95

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128×128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits (JSSC)*, 43(2):566–576. 45

[Likamwa et al., 2016] Likamwa, R., Hou, Y., Gao, Y., Polansky, M., and Zhong, L. (2016). RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision. *International Symposium on Computer Architecture (ISCA)*, pages 255–266. 14, 46

[Lin et al., 2020] Lin, Y., Larsson, V., Geppert, M., Kukelova, Z., Pollefeys, M., and Sattler, T. (2020). Infrastructure-based multi-camera calibration using radial projections. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 122

[Linan et al., 2002] Linan, G., Espejo, S., Dominguez-Castro, R., and Rodriguez-Vazquez, A. (2002). Architectural and basic circuit considerations for a flexible 128×128 mixed-signal SIMD vision chip. *Analog Integrated Circuits and Signal Processing*, 33(2):179–190. 46

[Lisondra et al., 2024] Lisondra, M., Kim, J., Murai, R., Zareinia, K., and Saeedi, S. (2024). Visual inertial odometry using focal plane binary features (bit-vio). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 17

[Liu et al., 2020] Liu, L., Gu, J., Lin, K. Z., Chua, T.-S., and Theobalt, C. (2020). Neural sparse voxel fields. *NeurIPS*. 16

[Liu et al., 2021] Liu, Y., Bose, L., Greatwood, C., Chen, J., Fan, R., Richardson, T., Carey, S. J., Dudek, P., and Mayol-Cuevas, W. (2021). Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array. *IET image processing*, 15(9):1883–1892. 14

[Luiten et al., 2024] Luiten, J., Kopanas, G., Leibe, B., and Ramanan, D. (2024). Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *3DV*. 16

[Lynch and Park, 2017] Lynch, K. M. and Park, F. C. (2017). *Modern robotics*. Cambridge University Press. 84

[Ma et al., 2018] Ma, Q., Goh, Z., Ruan, S., and Chirikjian, G. S. (2018). Probabilistic approaches to the AXB= YCZ AXB= YCZ calibration problem in multi-robot systems. *Autonomous Robots*, 42:1497–1520. 122

[Macenski et al., 2022] Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074. 114

[Marquardt, 1963] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441. 29

[Martel, 2019] Martel, J. N. (2019). *Unconventional processing with unconventional visual sensing: Parallel, distributed and event based vision algorithms & systems*. PhD thesis, ETH Zurich. 3

[Martel et al., 2015] Martel, J. N., Chau, M., Cook, M., and Dudek, P. (2015). Pixel interlacing to trade off the resolution of a cellular processor array against more registers. In *European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4. 14

[Martel et al., 2016] Martel, J. N., Müller, L. K., Carey, S. J., and Dudek, P. (2016). Parallel HDR tone mapping and auto-focus on a cellular processor array vision chip. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1430–1433. IEEE. 13, 14

[Martel et al., 2017] Martel, J. N., Müller, L. K., Carey, S. J., Müller, J., Sandamirskaya, Y., and Dudek, P. (2017). Real-time depth from focus on a programmable focal plane processor. *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, 65(3):925–934. 13, 14

[Martinec and Pajdla, 2007] Martinec, D. and Pajdla, T. (2007). Robust rotation and translation estimation in multiview reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 9

[Mateos et al., 2010] Mateos, G., Bazerque, J. A., and Giannakis, G. B. (2010). Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10):5262–5276. 9

[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 Conference on Computer Vision and Pattern Recognition (CVPR)*. 15, 17, 18, 61

[McConville et al., 2020] McConville, A., Bose, L., Clarke, R., Mayol-Cuevas, W., Chen, J., Greatwood, C., Carey, S., Dudek, P., and Richardson, T. (2020). Visual Odometry Using Pixel Processor Arrays for Unmanned Aerial Systems in GPS Denied Environments. *Frontiers in robotics and AI*, 7(126). 14, 47

[McCormac et al., 2017] McCormac, J., Handa, A., Davison, A., and Leutenegger, S. (2017). Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 11

[McEliece et al., 1998] McEliece, R. J., MacKay, D. J. C., and Cheng, J.-F. (1998). Turbo decoding as an instance of pearl's" belief propagation" algorithm. *IEEE Journal on selected areas in communications*, 16(2):140–152. 7

[McGann et al., 2023] McGann, D., Lassak, K., and Kaess, M. (2023). Asynchronous distributed smoothing and mapping via on-manifold consensus ADMM. *arXiv preprint arXiv.2310.12320*. 9, 95

[McMahan et al., 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR. 10

[Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 15

[Miluzzo et al., 2008] Miluzzo, E., Lane, N. D., Campbell, A. T., and Olfati-Saber, R. (2008). CaliBree: A self-calibration system for mobile sensor networks. In *Distributed Computing in Sensor Systems*, pages 314–331, Berlin, Heidelberg. Springer Berlin Heidelberg. 122

[Müller et al., 2022] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41(4):1–15. 16

[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 (T-RO)*, 31(5):1147–1163. 15, 64

[Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262. 44, 51, 52, 54, 71

[Murai, 2019] Murai, R. (2019). Visual odometry using a focal-plane sensor-processor. Master's thesis, Imperial College London. 44

[Murai et al., 2024] Murai, R., Alzugaray, I., Kelly, P. H., and Davison, A. J. (2024). Distributed simultaneous localisation and auto-calibration using gaussian belief propagation. *IEEE Robotics and Automation Letters*. 17, 19

[Murai et al., 2023a] Murai, R., Ortiz, J., Saeedi, S., Kelly, P. H., and Davison, A. J. (2023a). A robot web for distributed many-device localisation. *IEEE Transactions on Robotics*. 17, 19

[Murai et al., 2020] Murai, R., Saeedi, S., and Kelly, P. H. (2020). BIT-VO: Visual odometry at 300 fps using binary features from the focal plane. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. IEEE. 16, 18

[Murai et al., 2023b] Murai, R., Saeedi, S., and Kelly, P. H. (2023b). High-frame rate homography and visual odometry by tracking binary features from the focal plane. *Autonomous Robots*, pages 1–14. 17

[Murakami et al., 2015] Murakami, K., Yamakawa, Y., Senoo, T., and Ishikawa, M. (2015). Motion planning for catching a light-weight ball with high-speed visual feedback. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 339–344. IEEE. 3

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. 5, 30, 31, 32

[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*. 6, 38, 102

[Nabarro et al., 2023] Nabarro, S., van der Wilk, M., and Davison, A. J. (2023). Learning in deep factor graphs with gaussian belief propagation. *arXiv preprint arXiv:2311.14649*. 134

[Nagata and Sekikawa, 2023] Nagata, J. and Sekikawa, Y. (2023). Tangentially elongated gaussian belief propagation for event-based incremental optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21940–21949. 7

[Nesterov, 1983] Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547. 8

[Newcombe et al., 2011a] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 11, 15, 63

[Newcombe et al., 2011b] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011b). DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 11, 15

[Niemeyer et al., 2020] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 15

[Nießner et al., 2013] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3D Reconstruction at Scale using Voxel Hashing. In *Proceedings of SIGGRAPH*. 63

[Nistér, 2004] Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):0756–777. 51

[Ojala et al., 2002] Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):971–987. 49, 50

[Olson, 2011] Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. 113

[Olson et al., 2006] Olson, E., Leonard, J., and Teller, S. (2006). Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 8

[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*. 80, 93, 99

[Ortiz et al., 2022] Ortiz, J., Evans, T., Sucar, E., and Davison, A. J. (2022). Incremental Abstraction in Distributed Probabilistic SLAM Graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 7, 117, 128, 134

[Ortiz et al., 2020] Ortiz, J., Pupilli, M., Leutenegger, S., and Davison, A. J. (2020). Bundle adjustment on a graph processor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7, 12

[Papalia et al., 2023] Papalia, A., Fishberg, A., O'Neill, B. W., How, J. P., Rosen, D. M., and Leonard, J. J. (2023). Certifiably correct range-aided SLAM. *arXiv preprint arXiv:2302.11614*. 95

[Patwardhan and Davison, 2023] Patwardhan, A. and Davison, A. J. (2023). A distributed multi-robot framework for exploration, information acquisition and consensus. *arXiv preprint arXiv:2310.01930*. 7

[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. 7, 18, 118

[Pearl, 1982] Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. 6, 33

[Pineda et al., 2022] Pineda, L., Fan, T., Monge, M., Venkataraman, S., Sodhi, P., Chen, R. T., Ortiz, J., DeTone, D., Wang, A., Anderson, S., Dong, J., Amos, B., and Mukadam, M. (2022). Theseus: A Library for Differentiable Nonlinear Optimization. In *Neural Information Processing Systems (NeurIPS)*. 129

[Poikonen et al., 2009] Poikonen, J., Laiho, M., and Paasio, A. (2009). MIPA4k: A 64×64 cell mixed-mode image processor array. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1927–1930. 46

[Prisacariu et al., 2014] Prisacariu, V. A., Kähler, O., Cheng, M., Ren, C. Y., Valentin, J. P. C., Torr, P. H. S., Reid, I. D., and Murray, D. W. (2014). A framework for the volumetric integration of depth images. *arXiv preprint arXiv:1410.0925*. 15

[Ranganathan et al., 2007] Ranganathan, A., Kaess, M., and Dellaert, F. (2007). Loopy SAM. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 7

[Rebecq et al., 2016] Rebecq, H., Horstschäfer, T., Gallego, G., and Scaramuzza, D. (2016). EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600. 11

[Rhodes et al., 2022] Rhodes, C., Liu, C., and Chen, W.-H. (2022). Scalable probabilistic gas distribution mapping using gaussian belief propagation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9459–9466. IEEE. 7

[Rosen et al., 2019] Rosen, D., Carlone, L., Bandeira, A., and Leonard, J. (2019). SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. *International Journal of Robotics Research (IJRR)*, 38(2–3):95–125. 111

[Rosin, 1999] Rosin, P. L. (1999). Measuring corner properties. *Computer Vision and Image Understanding (CVIU)*, 73(2):291–307. 49

[Rosinol et al., 2020] Rosinol, A., Abate, M., Chang, Y., and Carlone, L. (2020). Kimera: an open-source library for real-time metric-semantic localization and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 11

[Rosinol et al., 2023] Rosinol, A., Leonard, J. J., and Carlone, L. (2023). Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE. 15, 64

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2564–2571. 56

[Salas-Moreno et al., 2013] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 11

[Sandström et al., 2023] Sandström, E., Li, Y., Van Gool, L., and R. Oswald, M. (2023). Point-slam: Dense neural point cloud-based slam. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 15, 64, 70, 71, 72, 73

[Schiff et al., 2009] Schiff, J., Sudderth, E. B., and Goldberg, K. (2009). Nonparametric belief propagation for distributed tracking of robot networks with noisy inter-distance measurements. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 7, 95

[Schöps et al., 2019] Schöps, T., Sattler, T., and Pollefeys, M. (2019). Surfelmeshing: Online surfel-based mesh reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 42(10):2494–2507. 63

[Schöps et al., 2019] Schöps, T., Sattler, T., and Pollefeys, M. (2019). Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 15, 63, 71

[Scona et al., 2022] Scona, R., Matsuki, H., and Davison, A. (2022). From scene flow to visual odometry through local and global regularisation in Markov random fields. *IEEE Robotics and Automation Letters*, 7(2):4299–4306. 7, 134

[Shiu and Ahmad, 1989] Shiu, Y. and Ahmad, S. (1989). Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB. *IEEE Robotics and Automation Letters*, 5(1):16–29. 122

[Solà et al., 2018] Solà, J., Deray, J., and Atchuthan, D. (2018). A micro Lie theory for state estimation in robotics. *arXiv:1812.01537*. 39, 66, 85, 86, 87, 88, 124

[Somasundaram et al., 2023] Somasundaram, K., Dong, J., Tang, H., Straub, J., Yan, M., Goesele, M., Engel, J. J., De Nardi, R., and Newcombe, R. (2023). Project Aria: A new tool for egocentric multi-modal ai research. *arXiv preprint arXiv:2308.13561*. 2

[Stahl, 2006] Stahl, S. (2006). The Evolution of the Normal Distribution. *Mathematics Magazine*, 79. 22

[Stauffert et al., 2020] Stauffert, J.-P., Niebling, F., and Latoschik, M. E. (2020). Latency and cybersickness: Impact, causes, and measures. a review. *Frontiers in Virtual Reality*, 1:582204. 3

[Stow et al., 2022a] Stow, E., Ahsan, A., Li, Y., Babaei, A., Murai, R., Saeedi, S., and Kelly, P. H. J. (2022a). Compiling cnns with cain: focal-plane processing for robot navigation. *Autonomous Robots*. 18

[Stow et al., 2022b] Stow, E., Murai, R., Saeedi, S., and Kelly, P. H. J. (2022b). Cain: Automatic code generation forÂ simultaneous convolutional kernels onÂ focal-plane sensor-processors. In *Languages and Compilers for Parallel Computing*, pages 181–197. Springer International Publishing. 18

[Strasdat et al., 2012] Strasdat, H., Montiel, J. M., and Davison, A. J. (2012). Visual SLAM: why filter? *Image and Vision Computing (IVC)*, 30(2):65–77. 5, 51

[Straub et al., 2019] Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H. M., Nardi, R. D., Goesele, M., Lovegrove, S., and Newcombe, R. (2019). The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*. 69

[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 *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. 52, 69, 103

[Sucar et al., 2021] Sucar, E., Liu, S., Ortiz, J., and Davison, A. J. (2021). iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 63, 64, 69, 70, 71, 72

[Suleiman et al., 2019] Suleiman, A., Zhang, Z., Carlone, L., Karaman, S., and Sze, V. (2019). Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. *IEEE Journal of Solid-State Circuits*, 54(4):1106–1119. 12

[Sun et al., 2022] Sun, C., Sun, M., and Chen, H. (2022). Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 16

[Sun et al., 2003] Sun, J., Zheng, N.-N., and Shum, H.-Y. (2003). Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(7):787–800. 7

[Sutter, 2005] Sutter, H. (2005). The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's journal*, 30(3):202–210. 10

[Sutter, 2011] Sutter, H. (2011). Welcome to the jungle. https://herbsutter.com/welcome-to-the-jungle. Accessed: 2024-01-20. 10

[Sze et al., 2017] Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329. 12

[Tang et al., 2024] Tang, J., Ren, J., Zhou, H., Liu, Z., and Zeng, G. (2024). Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *Proceedings of the International Conference on Learning Representations (ICLR)*. 16

[Tateno et al., 2017] Tateno, K., Tombari, F., Laina, I., and Navab, N. (2017). CNN-SLAM: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 11

[Taylor et al., 2006] Taylor, C., Rahimi, A., Bachrach, J., Shrobe, H., and Grue, A. (2006). Simultaneous localization, calibration, and tracking in an ad hoc sensor network. In *2006 5th International Conference on Information Processing in Sensor Networks*. 122

[Teed and Deng, 2021] Teed, Z. and Deng, J. (2021). DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. *Neural Information Processing Systems (NeurIPS)*, 34:16558–16569. 11, 64, 70, 71

[Teets and Whitehead, 1999] Teets, D. and Whitehead, K. (1999). The discovery of Ceres: How Gauss became famous. *Mathematics Magazine*, 72(2):83–93. 22

[Tian et al., 2022] Tian, Y., Chang, Y., Arias, F. H., Nieto-Granda, C., How, J. P., and Carlone, L. (2022). Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems. *IEEE Transactions on Robotics (T-RO)*, 38(4). 94

[Tian et al., 2021] Tian, Y., Khosoussi, K., Rosen, D. M., and How, J. P. (2021). Distributed certifiably correct pose-graph optimization. *IEEE Transactions on Robotics (T-RO)*, 37(6):2137–2156. 8, 94, 112

[Tian et al., 2020] Tian, Y., Koppel, A., Bedi, A. S., and How, J. P. (2020). Asynchronous and parallel distributed pose graph optimization. *IEEE Robotics and Automation Letters*, 5(4):5819–5826. 8, 94, 95, 111, 112

[Todescato et al., 2015] Todescato, M., Carron, A., Carli, R., and Schenato, L. (2015). Distributed localization from relative noisy measurements: a robust gradient based approach. In *European Control Conference (ECC)*. 95

[Toma et al., 2021] Toma, A.-I., Hsueh, H.-Y., Jaafar, H. A., Murai, R., Kelly, P. H., and Saeedi, S. (2021). Pathbench: A benchmarking platform for classical and learned path planning algorithms. In *2021 18th Conference on Robots and Vision (CRV)*, pages 79–86. IEEE. 18

[Triggs et al., 1999] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (1999). Bundle Adjustment — A Modern Synthesis. In *Proceedings of the International Workshop on Vision Algorithms, in association with ICCV*. 5, 29

[Vespa et al., 2018] Vespa, E., Nikolov, N., Grimm, M., Nardi, L., Kelly, P. H., and Leutenegger, S. (2018). Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151. 63

[Wan et al., 2017] Wan, J., Bu, S., Yu, J., and Zhong, L. (2017). Distributed simultaneous localization and mapping for mobile robot networks via hybrid dynamic belief propagation. *International Journal of Distributed Sensor Networks*, 13(8):1–19. 7, 95

[Wang et al., 2022a] Wang, A., Wang, P., Sun, J., Kortylewski, A., and Yuille, A. (2022a). Voge: a differentiable volume renderer using gaussian ellipsoids for analysis-by-synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 16

[Wang et al., 2023] Wang, H., Wang, J., and Agapito, L. (2023). Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 70, 71, 72

[Wang et al., 2014] Wang, J., Wu, L., Meng, M. Q.-H., and Ren, H. (2014). Towards simultaneous coordinate calibrations for cooperative multiple robots. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 122

[Wang et al., 2022b] Wang, Y., Wen, X., Yin, L., Xu, C., Cao, Y., and Gao, F. (2022b). Certifiably optimal mutual localization with anonymous bearing measurements. *IEEE Robotics and Automation Letters*, 7(4):9374–9381. 95

[Weiser, 1991] Weiser, M. (1991). The computer for the 21 st century. *Scientific american*, 265(3):94–105. 1

[Weiss and Freeman, 1999] Weiss, Y. and Freeman, W. (1999). Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Advances in neural information processing systems*, 12. 7, 38, 124, 129

[Whelan et al., 2015a] Whelan, T., Kaess, M., Johannsson, H., Fallon, M. F., Leonard, J. J., and McDonald, J. B. (2015a). Real-time large scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research (IJRR)*, 34(4-5):598–626. 15, 71

[Whelan et al., 2015b] Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., and Davison, A. J. (2015b). Elasticfusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*. 11, 15, 64

[Wu et al., 2024] Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., and Wang, X. (2024). 4d gaussian splatting for real-time dynamic scene rendering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 16

[Wymeersch et al., 2009] Wymeersch, H., Lien, J., and Win, M. Z. (2009). Cooperative localization in wireless networks. *Proceedings of the IEEE*, 97(2):427–450. 95

[Xiong et al., 2023] Xiong, Y., Varadarajan, B., Wu, L., Xiang, X., Xiao, F., Zhu, C., Dai, X., Wang, D., Sun, F., Iandola, F., et al. (2023). Efficientsam: Leveraged masked image pretraining for efficient segment anything. *arXiv preprint arXiv:2312.00863*. 16, 17

[Yang et al., 2022] Yang, X., Li, H., Zhai, H., Ming, Y., Liu, Y., and Zhang, G. (2022). Voxfusion: Dense tracking and mapping with voxel-based neural implicit representation. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 70, 71, 72, 73

[Yang et al., 2024] Yang, Z., Yang, H., Pan, Z., Zhu, X., and Zhang, L. (2024). Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *Proceedings of the International Conference on Learning Representations (ICLR)*. 16

[Yedidia et al., 2001] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Bethe free energy, kikuchi approximations, and belief propagation algorithms. *Advances in Neural Information Processing Systems*, 13(24). 6, 38

[Yi et al., 2024] Yi, T., Fang, J., Wu, G., Xie, L., Zhang, X., Liu, W., Tian, Q., and Wang, X. (2024). Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 16

[Zarándy, 2011] Zarándy, A. (2011). *Focal-plane sensor-processor chips*. Springer New York. 14, 46

[Zhang et al., 2017] Zhang, R., Zhu, S., Fang, T., and Quan, L. (2017). Distributed very large scale bundle adjustment by global camera consensus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29–38. 9

[Zhang et al., 2021] Zhang, Y., Hsiao, M., Dong, J., Engel, J., and Dellaert, F. (2021). MR-iSAM2: Incremental smoothing and mapping with multi-root Bayes tree for multi-robot SLAM. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 94

[Zhang, 1997] Zhang, Z. (1997). Parameter estimation techniques: A tutorial with application to conic fitting. In *Image and Vision Computing (IVC)*. 51, 83

[Zhu et al., 2024] Zhu, Z., Peng, S., Larsson, V., Cui, Z., Oswald, M. R., Geiger, A., and Pollefeys, M. (2024). Nicer-SLAM: Neural implicit scene encoding for RGB SLAM. *Proceedings of the International Conference on 3D Vision (3DV)*. 64

[Zhu et al., 2022] Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M. R., and Pollefeys, M. (2022). Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 64, 70, 71, 72, 73

[Zwicker et al., 2002] Zwicker, M., Pfister, H., van Baar, J., and Gross, M. (2002). Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238. 66

CHAPTER 9

Appendix

9.1 Derivation of Camera Pose Jacobian

Here, we detail how the camera pose Jacobian (Equation 4.6) is derived for Gaussian Splatting SLAM. Note that the derivation uses Lie theory and follows Section 2.6.6; however, we use the Left Jacobians on Lie groups Equation 4.5, since the camera is defined as $\mathbf{T}_{CW}$ in the camera coordinate not the world coordinate following graphics convention.

Firstly, we derive the derivative with respect to the centre of the Gaussians:

$$\frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\mathbf{T}_{CW}} = \lim_{\tau \to 0} \frac{\mathrm{Exp}(\tau) \cdot \boldsymbol{\mu}_C - \boldsymbol{\mu}_C}{\tau} \tag{9.1}$$

$$= \lim_{\tau \to 0} \frac{(\boldsymbol{I} + \tau^\wedge) \cdot \boldsymbol{\mu}_C - \boldsymbol{\mu}_C}{\tau} \tag{9.2}$$

$$= \lim_{\tau \to 0} \frac{\tau^\wedge \cdot \boldsymbol{\mu}_C}{\tau} \tag{9.3}$$

$$= \lim_{\tau \to 0} \frac{[\theta]_\times \boldsymbol{\mu}_C + \rho}{\tau} \tag{9.4}$$

$$= \lim_{\tau \to 0} \frac{-[\boldsymbol{\mu}_C]_\times \theta + \rho}{\tau} \tag{9.5}$$

$$= \begin{bmatrix} \boldsymbol{I} & -[\boldsymbol{\mu}_C]_\times \end{bmatrix} \tag{9.6}$$

where $\mathbf{T} \cdot \mathbf{x}$ is the group action of $\mathbf{T} \in \mathbf{SE}(3)$ on $\mathbf{x} \in \mathbb{R}^3$.

Similarly, we compute the Jacobian with respect to $\mathbf{W}$. Since the translational component is not involved, we only consider the rotational part $\mathbf{R}_{CW}$ of $\mathbf{T}_{CW}$.

$$\frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{R}_{CW}} = \lim_{\theta \to 0} \frac{\mathrm{Exp}(\theta) \circ \mathbf{W} - \mathbf{W}}{\theta} \tag{9.7}$$

$$= \lim_{\theta \to 0} \frac{(\boldsymbol{I} + \theta^\wedge) \circ \mathbf{W} - \mathbf{W}}{\theta} \tag{9.8}$$

$$= \lim_{\theta \to 0} \frac{\theta^\wedge}{\theta} \circ \mathbf{W} \tag{9.9}$$

$$= \lim_{\theta \to 0} \frac{[\theta]_\times}{\theta} \circ \mathbf{W} \tag{9.10}$$

Since skew-symmetric matrix is:

$$[\theta]_\times = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix} \tag{9.11}$$

The partial derivative of one of the component (*e.g.* θ_x) is:

$$\frac{\partial [\theta]_\times}{\partial \theta_x} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = [\mathbf{e}_1]_\times \tag{9.12}$$

where $\mathbf{e}_1 = [1, 0, 0]^\top, \mathbf{e}_2 = [0, 1, 0]^\top, \mathbf{e}_3 = [0, 0, 1]^\top$.

$$\frac{\partial \mathbf{W}}{\partial \theta_x} = [\mathbf{e}_1]_\times \mathbf{W} = \begin{bmatrix} \mathbf{0}_{1\times 3} \\ -\mathbf{W}_{3,:} \\ \mathbf{W}_{2,:} \end{bmatrix} \tag{9.13}$$

$$\frac{\partial \mathbf{W}}{\partial \theta_y} = [\mathbf{e}_2]_\times \mathbf{W} = \begin{bmatrix} \mathbf{W}_{3,:} \\ \mathbf{0}_{1\times 3} \\ -\mathbf{W}_{1,:} \end{bmatrix} \tag{9.14}$$

$$\frac{\partial \mathbf{W}}{\partial \theta_z} = [\mathbf{e}_3]_\times \mathbf{W} = \begin{bmatrix} -\mathbf{W}_{2,:} \\ \mathbf{W}_{1,:} \\ \mathbf{0}_{1\times 3} \end{bmatrix} \tag{9.15}$$

where $\mathbf{W}_{i,:}$ refers to the ith row of the matrix. After column-wise vectorisation of Equations 9.13 to 9.15, and stacking horizontally we get:

$$\frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{R}_{CW}} = \begin{bmatrix} -[\mathbf{W}_{:,1}]_\times \\ -[\mathbf{W}_{:,2}]_\times \\ -[\mathbf{W}_{:,3}]_\times \end{bmatrix}, \tag{9.16}$$

where $\mathbf{W}_{:,i}$ refers to the ith column of the matrix. Since the translational part is all zeros, with this we get Equation 4.6.