2022

Distributed Robotic Vision for Calibration, Localisation, and Mapping

Brendan James Halloran

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Distributed Robotic Vision for Calibration, Localisation, and Mapping

Brendan James Halloran

This thesis is presented as required for the conferral of the degree:

Doctor of Philosophy

Supervisor:
Dr. Prashan Premaratne

Co-supervisor:
Dr. Peter James Vial

The University of Wollongong
School of School of Electrical, Computer, and Telecommunication Engineering

August, 2021

This research has been conducted with the support of an Australian Government Research Training Program Scholarship.

Declaration

I, *Brendan James Halloran*, declare that this thesis is submitted in fulfilment of the requirements for the conferral of the degree *Doctor of Philosophy*, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Brendan James Halloran

November 10, 2022

Abstract

This dissertation explores distributed algorithms for calibration, localisation, and mapping in the context of a multi-robot network equipped with cameras and on-board processing, comparing against centralised alternatives where all data is transmitted to a singular external node on which processing occurs. With the rise of large-scale camera networks, and as low-cost on-board processing becomes increasingly feasible in robotics networks, distributed algorithms are becoming important for robustness and scalability. Standard solutions to multi-camera computer vision require the data from all nodes to be processed at a central node which represents a significant single point of failure and incurs infeasible communication costs. Distributed solutions solve these issues by spreading the work over the entire network, operating only on local calculations and direct communication with nearby neighbours.

This research considers a framework for a distributed robotic vision platform for calibration, localisation, mapping tasks where three main stages are identified: an initialisation stage where calibration and localisation are performed in a distributed manner, a local tracking stage where visual odometry is performed without inter-robot communication, and a global mapping stage where global alignment and optimisation strategies are applied. In consideration of this framework, this research investigates how algorithms can be developed to produce fundamentally distributed solutions, designed to minimise computational complexity whilst maintaining excellent performance, and designed to operate effectively in the long term. Therefore, three primary objectives are sought aligning with these three stages.

The first goal is the development of a robust calibration and localisation algorithm for the initialisation of the multi-robot platform. Two methods are proposed. The first utilises one-dimensional calibration performed locally at each node to establish the focal lengths and other internal camera parameters, then the 3D points observed in the world are aligned between nodes to construct a vision graph and perform neighbourhood-based optimisation, and finally the local estimates of relative localisation are brought into global consensus with a message-passing algorithm. Evaluations demonstrate that it has comparable performance to the equivalent centralised algorithm. The second method addresses shortcomings in the prior method:

constraints on the motion of the calibration object and the validity of the global objective function. An improved algorithm for general-motion multi-camera one-dimensional calibration and localisation is proposed with better numerical conditioning and simplified nonlinear optimisation. Alternating direction method of multipliers is employed to separate the global optimisation objective into local objectives that can be aligned using a message-passing algorithm. Experimental evaluation on synthetic and real data demonstrates high accuracy in centralised and distributed networks, showing superior performance in estimation of focal lengths and extrinsic parameters.

The second goal is the development a visual odometry method that is suitable for use in the second stage, aiming to be highly accurate, computationally efficient, and utilising a data-type suitable for the inter-robot pose estimates required for the third stage. Direct methods in visual odometry are highly accurate and efficient, however, not suitable for wide-baseline matching where they instead rely on the robust feature descriptors used in indirect methods. A hybrid direct-indirect approach is proposed that incorporates sparsely-evaluated binary feature descriptors of keypoints into the direct visual odometry pipeline using a pseudo-gradient based on Hamming weights. The performance of this algorithm is evaluated on real data, with results showing that the primary method improves significantly upon similar methods. The final goal is to address the convergence speed and accuracy for distributed algorithms for use in the first and third stage of the framework. Gaussian belief propagation is an effective message-passing algorithm for performing inference on graphs, providing approximate inference on graphs with cycles. It has been shown that faster convergence using edge-weights improves accuracy, however, existing methods for edge-weight selection rely on global information. An empirical method for edge-weight selection based on node degree is proposed for small-world networks. This method is evaluated on a simplified localisation problem and demonstrates greatly improved convergence speed and accuracy, with more significant improvements seen in larger networks.

Evaluations of the proposed methods show that accuracy at these various stages of the pipeline have been improved, with the distributed algorithms showing similar performance to centralised counterparts and improved performance compared to prior methods. This validates the pipeline as a basis for a distributed robotic vision platform capable of calibration, localisation, and mapping.

Acknowledgements

There are a huge number of people that I would like to thank for their endless support over the years. First and foremost, I would like to thank Dr Prashan Premaratne for his guidance and advice towards my work. It is his encouragement to pursue this degree that has led me to this moment. Furthermore, the School of Electrical, Computer, and Telecommunication Engineering and the entire University of Wollongong have been tremendously supportive over the years. I was provided with a welcoming and supportive community through my time working on this degree. I would also like to thank my family and, in particular, my parents who gave me so much reinforcement over the years and provided endless proofreading. Finally, I would like to thank Bec for her encouragement, even when things were most difficult.

Contents

List of figures

List of tables

List of abbreviations and symbols

Abbreviations

ADMM	Alternating direction method of multipliers
BA	Bundle adjustment
BCA	Brightness consistency assumption
BP	Belief propagation
BRIEF	Binary robust independent elementary features
BRISK	Binary robust invariant scalable keypoints
CSN	Camera sensor network
DCA	Descriptor consistency assumption
DIAC	Dual image of the absolute conic
EAP	Edge-appearance probability
EKF	Extended Kalman filter
FABMAP	Fast appearance-based mapping
FAST	Features from accelerated segment test
FC-LK	Forwards-compositional Lucas-Kanade
FOV	Field-of-view
GaBP	Gaussian belief propagation
GBP	Generalised belief propagation
IAC	Image of the absolute conic
IC-LK	Inverse-compositional Lucas-Kanade
ICP	Iterative closest point
IMU	Inertial measurement unit

KF	Kalman filter
KLT	Kanade-Lucas-Tomasi
LK	Lucas-Kanade
LLS	Linear least-squares
LM	Levenberg-Marquardt
MAP	Maximum *a posteriori*
ML	Maximum likelihood
MMSE	Minimum mean-squared error
MRF	Markov random field
NBP	Non-parametric belief propagation
NLLS	Nonlinear least-squares
ORB	Oriented FAST and rotated BRIEF
PGO	Pose-graph optimisation
PnP	Perspective-n-point
RANSAC	Random sample consensus
RGB-D	Red, blue, green, depth
RMSE	Root-mean-squared error
SfM	Structure from motion
SIFT	Scale-invariant feature transform
SLAM	Simultaneous localisation and mapping
SURF	Speeded up robust feature
TRW-NBP	Tree-reweighted non-parametric belief propagation
VIO	Visual inertial odometry
VO	Visual odometry

Symbols

$b_i^{(t)}(\cdot)$	The belief at node i for time t
$\mathcal{E}$	The edges in an undirected graph
$\mathcal{G}$	An undirected graph

∇I	The image gradient
$\mathbf{J}$	A Jacobian matrix
$\mathbf{K}$	The intrinsic camera matrix
$\mathbf{m}$	A 2D image point
$m_{i \to j}^{(t)}(\cdot)$	A message from node i to node j at time t
$\mathbf{M}$	A 3D world point
$\mathcal{M}$	A projective reconstruction of a 3D world point
$\mathcal{N}_i$	The neighbours of node i
P	The inverse-variance of a GaBP message or belief
$\mathbf{P}$	A camera matrix
$\mathcal{P}$	A projective matrix
$\mathcal{Q}$	The mapping of components between vectors
$\mathbf{r}$	A residual vector
$\mathbf{R}$	A rotation matrix $\in SO(3)$
$\mathbf{S}$	A normalisation scaling matrix
$\mathbf{t}$	A translation vector
$\mathbf{T}$	A transformation matrix $\in SE(3)$
u_0	Camera principal point x coordinate
v_0	Camera principal point y coordinate
$\mathcal{V}$	The vertices in an undirected graph
$w(\cdot)$	A warp function
$\mathbf{x}$	A measurement or estimation vector
$\mathbf{y}$	The Lagrangian multiplier vector
$\mathbf{z}$	A random vector being estimated
α	Focal length along the x-axis
β	Focal length along the y-axis
γ	Skew between optical axes
μ	The mean of a GaBP message or belief
ξ	The Lie-algebra representation of a pose $\in \mathfrak{se}(3)$

$\pi(\cdot)$	The camera projection function
ρ	A message weight
Σ	Variance
$\phi_i(\cdot)$	The node potential for node i
$\mathbf{\Phi}$	A feature descriptor
$\psi_{ij}(\cdot)$	The edge potential between node i and j
ω	The image of the absolute conic
ω^*	The dual image of the absolute conic

Chapter 1

Introduction

Contents

1.1 Overview

Distributed computer vision algorithms are a novel approach to making large camera sensor networks (CSNs) and robotics platforms more robust and scalable. A CSN is a spatially distributed network of camera nodes which fuse their image data such to provide a number of advantages over single-view computer vision. CSNs can be deployed over a large area for wider coverage that could be required for disaster response and security applications, and they also can provide multiple viewpoints which can be exploited for three-dimensional scene analysis and gives redundancy against occlusion [1]. Traditional approaches to computer vision on CSNs utilise centralised algorithms which rely on a central node to process data from all other nodes [1, 2]. In battery powered and wireless applications it can be difficult and expensive to communicate with a central node, particularly in larger systems [2]. Additionally, this central node bears the risk of a single point of failure [3]. This promotes the need for distributed algorithms, where a network of functionally identical nodes perform local computations that can be incorporated into a global consensus without the need for any centralised processing.

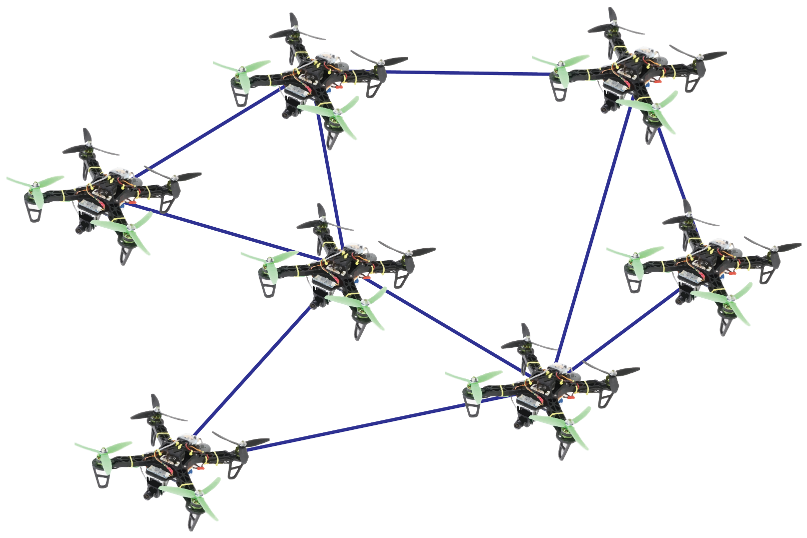


Figure 1.1: A distributed robotic vision network represented as an undirected graph, with robots as nodes and edges corresponding to communication capabilities.

With the increasing popularity of *unmanned aerial vehicles*, as well as the reducing cost of powerful single-board computers such as the Raspberry Pi, these distributed algorithms are being increasingly desirable for robotic vision applications. A single vision-enabled robot can perform tasks such as calibration and localisation [4], which then enables higher-level tasks such as visual odometry or *simultaneous localisation and mapping* (SLAM) that can be utilised for the exploration and mapping of an unknown environment [5]. With the addition of more robots, the exploration becomes collaborative if the robots are able to share a common understanding about the world and their collective goals [6]. As with the general case of CSNs, centralised approaches to these multi-robot tasks results in a fragile system. Therefore, distributed robotic visual algorithms are crucial for the development of advanced multi-robot platforms and applications. Consider the network of vision-enabled robot nodes given in Figure 1.1, where the ability to communicate between nodes is represented by an undirected graph [1–3, 7]. Each node is unable to communicate with all other nodes, instead only having a small number of direct neighbours. Therefore, the implementation of any robotic vision tasks must be done by combining local processing at the nodes and communication with those direct neighbours. Using such an approach, algorithm for multi-robot calibration [7], localisation [8], visual odometry, and SLAM can be implemented [9].

Calibration and localisation are the crucial first steps for CSNs which then enables the many higher-level tasks to be done and involves using constraints on the scene to determine the camera model parameters and the relative poses between cameras [4]. Calibration comes in many different formats which have strengths and weaknesses based on their use cases. Calibration algorithm can be classified based on the calibration object that they use: 3D [10–12], 2D [13], 1D [14, 15], or self-calibration [16–18]. 2D calibration is highly flexible in most cases, however, in multi-camera situations the calibration object can self-occlude at wide angles and therefore 1D calibration and self-calibration can be more appealing. Localisation is generally done in tandem with the calibration procedure as the information from the calibration object reveals this pose information, however, once the camera model is determined localisation can also be performed solely based on arbitrary correspondences between cameras [19]. Specifically for the case of calibrating and localising distributed CSNs, most approaches utilise self-calibration due to the flexibility it has in viewing angles of each camera [7, 8].

Once the calibration has been established, higher-level tasks such as visual odometry and SLAM can be be performed. Visual odometry is an iterative process of estimating pose information between subsequent images from the camera as the robot moves throughout a scene, then using those poses to reconstruct a map of the scene [5]. These two processes are called tracking and mapping [20]. SLAM expands upon this for long-term operation by identifying when a location has been revisited and relocalising when tracking is lost [21–23]. There are a wide array of methodologies for SLAM, however, the type considered most closely in this work is keyframe-SLAM, where images are periodically elevated to 'keyframe' status and information is accumulated in those keyframes to improve tracking accuracy and efficiency [20]. For distributed multi-robot SLAM systems, the most common approach is to build upon the underlying single-robot SLAM systems by adding features of map alignment and merging [6, 24] as well as global optimisation [25–27] that are implemented using a range of distributed processing algorithms [2].

Research in the field reveals a common trend of applying distributed algorithms to a consistent pipeline of calibration, localisation, and SLAM to produce robust distributed robotic vision systems [9, 28].

The remaining sections of this chapter are organised as follows: Section 1.2 presents the motivation and problems to be addressed, Section 1.3 asks the research questions that have arisen from the motivation and open problems, Section 1.4 states the objectives for the work addressing these questions and problems, Section 1.5 outlines the structure of the thesis, and finally Section 1.6 details the publications that have been produced over the course of this study.

1.2 Research motivation and problems

Distributed algorithms have been successfully applied to many problems such as flocking [29], formation control [30], and sensor fusion [31]. Much work over the last decade has also gone into adapting distributed algorithms to computer vision tasks, such as camera calibration [7], localisation [8], and visual tracking [32]. The importance of large-scale CSNs has been noted by many researchers, particularly in security and surveillance in buildings and public spaces, disaster response, robot coordination, and mapping of large environments [1, 5, 32, 33]. Radke states that in these large CSNs "the sheer number of cameras now feasible in a visual sensor network may preclude the use of a centralized algorithm" [3]. Song et al. write that there are additional problems with centralised algorithms including bandwidth constraints and security concerns [1]. They write, "...the cameras would have to act as autonomous agents... At the same time, however, the decisions of the cameras need to be coordinated so that there is a consensus about the task." Olfati-Saber identified the importance of these algorithms being robust to changing topologies and time delays [34]. Considering a network of mobile agents, he states "Since the nodes of the network are moving, ... communication links can fail simply due to the existence of an obstacle... [or] new links between nearby agents are created because the agents come to an effective range of detection." Furthermore, realistic wireless networks will have time-delays between the communicating nodes and therefore need to be engineered so as not to be fragile to these delays [35]. With regard to limitations on communication and computational complexity, particularly for SLAM, Cadena et al. [5] identified that, "[a] relatively unexplored issue is how to adapt existing SLAM algorithms to the case in which the robotic platforms have severe computational constraints. This problem is of great importance when the size of the platform is scaled down, e.g., mobile phones, micro aerial vehicles, or robotic insects. Many SLAM algorithms are too expensive to run on these platforms."

Reviewing the significant trends in the application of distributed algorithms to these robotic vision tasks, a framework for a multi-robot system performing SLAM can be identified as adhering to the three stages given in Figure 1.2. Firstly, an initialisation is performed to establish the calibration and initial localisation, then local visual odometry is performed at each node without the need to involve other robots, and finally global alignment and optimisation of the map is performed. Distributed algorithms are used in the first and third stages, whilst the second stage operates in the same manner as single-robot algorithms.

From this, a number of open problems and issues with current solutions can be identified:

- Existing distributed calibration techniques almost exclusively rely on self-calibration, which is not able to determine the scale of the scene for its localisation.

- Many distributed calibration and localisation techniques either do not involve the intrinsic parameters or do no optimise over an easily identifiable global

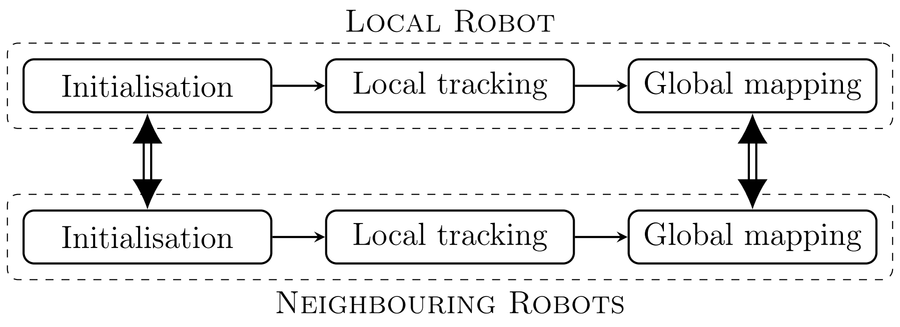


Figure 1.2: The three main stages of a distributed multi-robot system performing calibration, localisation, visual odometry and SLAM. Inter-robot communication is done during the first and third stages where distributed algorithms are applied.

objective for the full set of parameters.

- Distributed SLAM systems rely on *indirect* SLAM methodologies for the local tracking and do not take advantage of computationally efficient direct methodologies.
- Existing *direct* SLAM methodologies do not natively rely on data-types suitable for inter-robot alignment and so compute such data solely for wide-baseline matching, which can be seen as wasteful use of computational resources.
- Convergence speed and accuracy is crucial for distributed algorithms, however, many techniques for accelerating convergence and improving accuracy rely on global information about the network that would not necessarily be known at a given robot node.

1.3 Research questions

This research aims to improve the applicability of distributed computer vision to robotic networks with on-board processing in the undertaking of calibration, localisation, and SLAM by addressing the following questions:

Firstly How can a robotic camera sensor network perform and maintain calibration, localisation, and higher-order tasks such as SLAM in an entirely distributed manner?

Secondly How can these algorithms be made to operate effectively within the constraints of low-cost on-board processing?

Thirdly Can a consistent framework be established for a distributed SLAM system from initialisation to long-term operation?

In addressing the first question, this study will analyse **(a)** how centralised algorithms can be *adapted* to distributed networks, and **(b)** how the tasks can be re-framed to produce *fundamentally distributed* solutions. Also, in response to the second question, this dissertation explores **(c)** how algorithms can be made to minimise the computational complexity required for each local node whilst **(d)** maintaining robustness both locally and over the entire network. Finally, to address the third question, this study will evaluate **(e)** distributed solutions to various stages of a calibration, localisation, and SLAM pipeline as well as **(f)** local single-node solutions to aspects of the pipeline that support the distributed components.

1.4 Research objectives

These questions and problems give rise to a number of objectives that contribute towards performing calibration, localisation, and SLAM on a distributed robotic vision system. The primary objective is to investigate algorithms relating to this overall pipeline, enhancing the ability for robotic vision networks to complete those key tasks. To answer those stated questions, the following objectives have been accomplished:

- The investigation of research trends in the areas of calibration, localisation, visual odometry, and SLAM with a focus on the applicability of various algorithms to distributed systems.
- The development of calibration and localisation algorithms that are suitable for multi-camera and distributed systems.
- The development of a fundamentally distributed localisation algorithm that can serve as a highly accurate initialisation for a multi-robot SLAM system, providing known scale.
- The adaption of the direct visual odometry process to robust feature descriptors to produce a hybrid method that is efficient for low-cost hardware whilst maintaining the ability for wide-baseline matching.
- The improvement of direct visual odometry methods with regard to their robustness to lighting changes.
- The development of a method for choosing edge-weights in an undirected graph that improve convergence of Gaussian belief propagation in small-world networks.
- The application of accelerated Gaussian belief propagation to localisation and pose graph optimisation for improved accuracy.

1.5 Structure of the dissertation

The remaining sections of the dissertation are organised into the following chapters:

Chapter 2 provides an overview of research in the areas of calibration, localisation, visual odometry and SLAM for robotic vision platforms, with particular consideration for how distributed algorithms can be used to solve these problems for a network of robots. The various types of calibration and localisation are considered by the calibration objects that are utilised and how they can be applied to multi-camera systems. The related problems of visual odometry and SLAM are discussed with relation to a generalised keyframe-SLAM framework and the competing paradigms for solving the various sub-problems. Then, three groups of distributed algorithms are discussed and related to calibration, localisation, visual odometry and SLAM. The major trends and open problems are identified and related to the subsequent work presented in this dissertation.

Chapter 3 presents a method for distributed calibration and localisation of a camera sensor network based on one-dimensional calibration and Gaussian belief propagation. This method calibrates each camera node locally based on observations of a 1D calibration object which are observed at all nodes. The point-cloud of estimated world points at each node is then used to initialise a lattice-structured graph for neighbourhood-based bundle adjustment, then the localisation estimates are brought into global consensus using Gaussian belief propagation.

Chapter 4 improves upon the method presented in Chapter 3 by performing a general-motion one-dimensional calibration and localisation across all nodes and producing a global bundle adjustment, rather than a neighbourhood-based one, using the alternating direction method of multipliers applied to a small-world network with the distributed averaging step performed using Gaussian belief propagation.

Chapter 5 presents a method for single-robot visual odometry that has been designed to take advantage of the computation efficiencies of direct methods coupled with the ability to perform wide-baseline matching that is provided by the robust feature descriptors of indirect methods. This is done by performing spare calculation of binary descriptors which are then used in direct image alignment based on a pseudo-gradient. The purpose of this is to provide a visual odometry method that is suitable for the low-powered hardware that would be used in a distributed robotic vision network whilst still utilising the data-types suitable for multi-robot SLAM.

Chapter 6 analyses the problem of determining edge-weights for Gaussian belief propagation in a manner that only utilises local information. This is considered for small-world networks, which are a common naturally occurring network type that can be seen as an interpolation between random graphs and lattice graphs. An empirical method is derived for optimal edge-weights based on the average node-degree across the network as well as the local node-degree at a given node. This is then shown to improve convergence speed and accuracy for a simplified version of the multi-robot localisation problem.

Chapter 7 summarises the propose methodologies and considers their relation to the overall framework of performing visual SLAM on a distributed multi-robot network. Future research directions are also discussed.

1.6 Publications

The publications outlined below are the direct result of the research presented in this dissertation done as a part of the Doctor of Philosophy program undertaken at the University of Wollongong:

- B. Halloran *et al.*, 'Distributed one dimensional calibration and localisation of a camera sensor network,' in *Proceedings of ICIC 2017: Intelligent Computing Theories and Application*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2017, pp. 581–593, ISBN: 978-3-319-63312-1. DOI: `10.1007/978-3-319-63312-1_51`.
  This paper presents a distributed algorithm for the calibration and localisation of a camera sensor network. Robust calibration is performed at each node using a one-dimensional calibration object consisting of collinear points moving about a single fixed point. Next, each node builds a vision graph and performs cluster-based bundle adjustment utilising the 3D points of the calibration object. Finally, the local estimates are brought to global consensus using Gaussian belief propagation. The benefit of this method is that it provides a flexible and accurate method of metric calibration and localisation to known scale for a distributed network of cameras.

- B. Halloran *et al.*, 'Optimizing edge weights for distributed inference with gaussian belief propagation,' in *Proceedings of ICIC 2018: Intelligent Computing Theories and Application*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2018, pp. 46–59, ISBN: 978-3-319-95930-6. DOI: `10.1007/978-3-319-95930-6_6`.
  This paper analyses the process of using Gaussian belief propagation for performing inference and distributed averaging on a distributed sensor network.

Gaussian belief propagation only provides approximate values when used on cyclic graphical structures, as is the likely case in a robotic network, however, faster convergence generally leads to more accurate results. Choosing appropriate edge-weights for fast convergence often relies on global information on the structure of the network. This research investigated an empirical method for determining edge-weights using only local information, specifically for small-world networks. This leads to faster and more accurate convergence.

- B. Halloran *et al.*, 'Single and multi-channel direct visual odometry with binary descriptors,' in *Proceedings of ICIC 2019: Intelligent Computing Methodologies*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2019, pp. 86–98, ISBN: 978-3-030-26766-7. DOI: `10.1007/978-3-030-26766-7_9`.
  This paper presents a method for robust visual odometry that incorporates binary descriptors into the direct image alignment pipeline through a pseudo-gradient based on the Hamming weights of the descriptors. This produces a direct method that is more robust to lighting changes in the scene. Furthermore, although the computation of descriptors is more computationally expensive than operating on raw image values, as is the normal procedure in direct methods, this method still has the computational savings of avoiding feature descriptor matching. Instead, the method results in implicit correspondences.

- B. Halloran *et al.*, 'Robust one-dimensional calibration and localisation of a distributed camera sensor network,' *Pattern Recognition*, vol. 98, 107058, pp. 1–12, 2020. DOI: `10.1016/j.patcog.2019.107058`.
  This paper extended the previous work looking at calibration and localisation with one-dimensional objects in a distributed camera sensor network. A general-motion multi-view one-dimensional calibration methods was used to initialise local estimates at each camera node, then a vision graph was initialised to a small-world network structure. Finally, a combination on alternating direction method and multipliers and Gaussian belief propagation was used to perform a distributed global bundle adjustment across all nodes. This method improved upon the prior method by relaxing constraints on the calibration object and ensuring that the final optimisation had a consistent global objective function.

In addition, the following joint publications list contributions made during the undertaking of the Doctor of Philosophy program:

- I. J. Kadhim *et al.*, 'A comparative analysis among dual tree complex wavelet and other wavelet transforms based on image compression,' in *Proceedings of*

ICIC 2017: Intelligent Computing Theories and Application, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2017, pp. 569–580, ISBN: 978-3-319-63312-1. DOI: `10.1007/978-3-319-63312-1_50`

- Q. Al-Shebani *et al.*, 'Co-simulation method for hardware/software evaluation using xilinx system generator: A case study on image compression algorithms for capsule endoscopy,' in *Proceedings of 2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, IEEE, 2018, pp. 1–4. DOI: `10.1109/ICSPCS.2018.8631737`
- C. Shiranthika *et al.*, 'Realtime computer vision-based accurate vehicle counting and speed estimation for highways,' in *Proceedings of ICIC 2019: Intelligent Computing Methodologies*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2019, pp. 583–592, ISBN: 978-3-030-26766-7. DOI: `10.1007/978-3-030-26763-6_56`
- I. J. Kadhim *et al.*, 'Comprehensive survey of image steganography: Techniques, evaluations, and trends in future research,' *Neurocomputing*, vol. 335, pp. 299–326, 2019, ISSN: 0925-2312. DOI: `10.1016/j.neucom.2018.06.075`
- P. Premaratne *et al.*, 'Optimization of low-speed dual rotor axial flux generator design through electromagnetic modelling and simulation,' in *Proceedings of ICIC 2021: Intelligent Computing Theories and Application*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2021, pp. 786–801, ISBN: 978-3-030-84522-3. DOI: `10.1007/978-3-030-84522-3_64`

Chapter 2

Literature review

Contents

2.1 Summary of contributions

- Reviews state-of-the-art algorithms applicable to each stage of the multi-robot framework discussed in this thesis, highlighting research gaps and open problems, and applies those findings to the given framework.

- Thoroughly reviews the calibration and localisation problem for single cameras and multiple cameras, discussing solutions according to object-type and the related distortion models involved. Then, identifies the strengths and weaknesses of various calibration methods in their application to the initialisation of the multi-robot framework.

- Presents a generalised framework for visual odometry and SLAM, reviewing how a wide range of algorithms fit into this framework whilst utilising different paradigms and technologies. Then, identifies how these trends suit the local and global mapping steps of the multi-robot framework, as well as the research gaps in each paradigm.

- Explores how calibration, localisation, and SLAM algorithms are extended to distributed robotics platforms through the graphical interpretation of the network and the use of distributed algorithms, and examines the open problems in adapting these algorithms to be distributed in nature.

2.2 Introduction

This chapter provides an overview of the state-of-the-art research and trends in the robotic vision areas of calibration, localisation, visual odometry, and SLAM. Of particular interest is the multi-robot case where graphical methods can be used to produce distributed consensus algorithms that solve these challenging problems, following the multi-robot framework shown in Figure 1.2.

Calibration is the necessary initialisation step for many vision applications which enables metric understanding of the scene. In the single camera case this involves the intrinsic parameters of the camera such as focal length and principle point, where as in the multi-camera case the extrinsic parameters or localisation becomes much more important. The problems of calibration and localisation can be solved using special calibration objects that are placed in the scene or by leveraging constraints on the scene itself. For robotic vision, localisation provides an initial state of the robot and the intrinsic calibration allows the relationships between views of the robot to be understood.

Mobile robotics platforms rely on the more challenging localisation problems of visual odometry and SLAM. These are two closely related problems of continuous localisation and scene reconstruction. Although the calibration and localisation problem can provide an initial state for the robot or robots, localisation needs to be updated as a robot moves throughout the scene. Visual odometry solves this using static constraints of the scene to determine motion between views. For SLAM, the algorithms of visual odometry are robustified by relying more heavily on the reconstruction of the scene to improve localisation, detect when locations are revisited, or to detect and correct localisation failures.

These algorithms can be extended to distributed multi-robot scenarios, where each robot works with its neighbours to achieve a globally consistent understanding of the scene without the use of a central processor. This involves considering the underlying algorithms, such as average consensus and belief propagation, exploring graphical representations of camera sensor networks, and reviewing how this has been used to produce distributed approaches to the four key problems.

2.2.1 Motivation

The motivation of the literature review presented in this chapter is to is to analyse the wide range of algorithms that can be used to implement the various stages of the

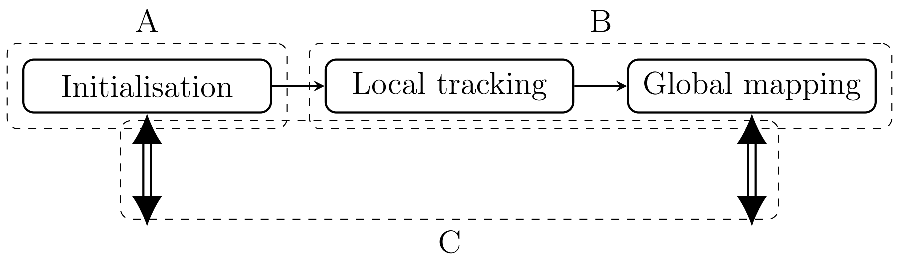


Figure 2.1: The three main stages of a distributed multi-robot system performing calibration, localisation, visual odometry and SLAM. Inter-robot communication is done during the first and third stages where distributed algorithms are applied.

distributed multi-robot calibration, localisation, visual odometry, and SLAM framework given in Figure 1.2. From the point of view of a single robot, this framework is related to the three main sections of this chapter is Figure 2.1. Part A, relating to the initialisation, is discussed through an analysis of calibration and localisation algorithms; part B, covering the local tracking and global mapping components, relates to the discussion on visual odometry and SLAM algorithms; and part C, involving the inter-robot communication, is discussed through an analysis of distributed algorithms and their relationship to robotic vision.

2.2.2 Organisation

Based on these motivations, this chapter is broken into three main sections — calibration and localisation, visual odometry and SLAM, and multi-robot distributed algorithms. Calibration and localisation are discussed in Section 2.3. In this section, the general calibration problem statement is introduced, as well as the different classes of calibration algorithms ranging from the use of three-dimensional objects to self-calibration. The strengths and weaknesses of these algorithms are then discussed in relation to their use in multi-camera systems. The more challenging problems of visual odometry and SLAM are discussed in detail in Section 2.4. Fristly, the core paradigms of VO and SLAM are introduced, then the many state-of-the-art algorithms are categorised according to these paradigms, and finally they are analysed according to a generalised keyframe SLAM architecture. Section 2.5 applies the problems of multi-robot calibration, localisation, visual odometry, and SLAM to distributed networks. This is done by introducing the graphical representation of camera sensor networks and the core underlying algorithms for distributed processing, then exploring how these have been applied to the present problems. Finally, Section 2.6 ties these areas of research together and discusses how this work has influenced the novel work presented in this dissertation. This is explored through the presentation of a framework for a distributed multi-robot vision system capable of calibration, initial localisation, and SLAM.

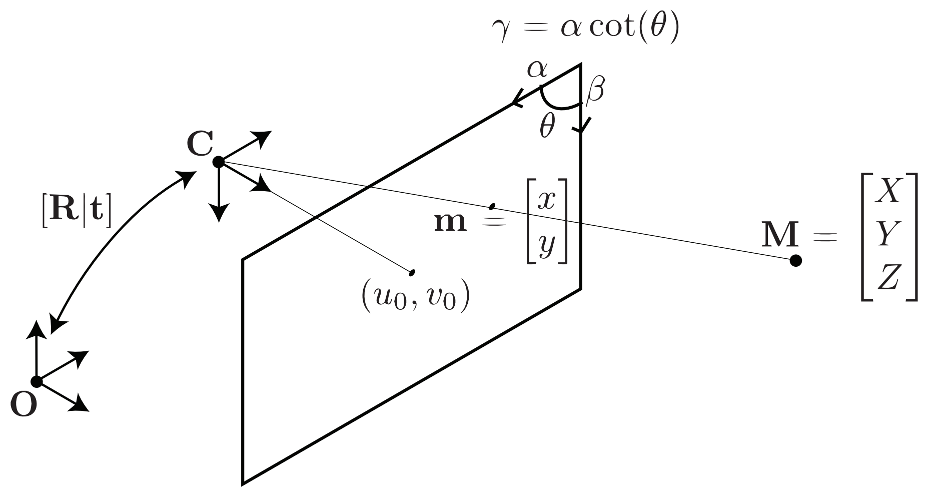
2.3 Calibration and localisation

Calibration is the crucial first step in vision systems which allows metric 3D information to be attained from images. In the single-camera case, calibration aims to determine the intrinsic parameters of the camera which relate to the focal length, principal point, skew, and lens distortion inherent to the camera. This process might also localise the camera's position and orientation with respect to some known points in the scene, known as its extrinsic parameters. In the multi-camera case, individual intrinsic calibrations are sought for each camera as well as all extrinsic parameters that define the cameras' positions and orientations relative to each other. This calibration and localisation process provides an initial state that is crucial in higher-order vision tasks, with visual odometry and visual SLAM being of particular interest in this chapter.

This section reviews the major work in the field of calibration and localisation and its application to distributed camera systems, with the motivation of finding suitable algorithms for the first component identified in Figure 2.1. Section 2.3.1 introduces the calibration and localisation problem statement. Section 2.3.2 discusses the basic algorithms that are employed – 3D calibration, 2D calibration, 1D calibration, and self-calibration. These four categories of algorithms have different complexities, strengths, weaknesses, and use-cases which inform the situations that they are often used. Section 2.3.3 expands on this information and looks at how these calibration paradigms are applied to multi-camera systems, both stereo camera pairs with known baselines as well as n-view systems with unknown extrinsic parameters. It is in these multi-camera systems where the localisation problem becomes fundamentally a part of the overall calibration. Finally, Section 2.3.4 summarises these findings and highlights the open problems related to this thesis.

2.3.1 Calibration problem statement

The goal of camera calibration and localisation is to determine the parameters that define how a point in 3D space is projected into the image plane of a camera to produce a pixel coordinate or 2D point. A standard camera is usually modelled using the pinhole camera model, given in Figure 2.2. In this model a point in space is denoted as $\mathbf{M} = [X, Y, Z]^T$ and its image is $\mathbf{m} = [x, y]^T$, which are sometimes given in homogeneous form as $\tilde{\mathbf{M}} = [X, Y, Z, 1]^T$ and $\tilde{\mathbf{m}} = [x, y, 1]^T$ respectively. The image point $\mathbf{m}$ is given by the intersection of the optical ray between the 3D point $\mathbf{M}$ and the camera centre $\mathbf{C}$ with the image plane. To compute this intersection, first the 3D point is transformed from the world coordinate frame into the camera frame by the extrinsic parameters of rotation, $\mathbf{R} \in \mathrm{SO}(3)$, and translation, $\mathbf{t} \in \mathbb{R}^3$. This transformed 3D point is then projected into the image plane according to



$\gamma = \alpha \cot(\theta)$
α
β
θ
$\mathbf{C}$
$[\mathbf{R}|\mathbf{t}]$
$\mathbf{m} = \begin{bmatrix} x \\ y \end{bmatrix}$
$\mathbf{M} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
(u_0, v_0)
$\mathbf{O}$

Figure 2.2: The pinhole camera model, where a 3D world point $\mathbf{M}$ is projected to 2D image point $\mathbf{m}$ according to intrinsic parameters α, β, and (u_0, v_0), as well as extrinsic parameters $\mathbf{t}$ and $\mathbf{R}$.

the intrinsic parameters of the x- and y-axis focal lengths α and β, the principal point (u_0, v_0), and the skew between the image axes, γ. This transformation and projection is given in Equation 2.1, with the intrinsic parameter matrix $\mathbf{K}$ being defined in Equation 2.2. In modern cameras, the skew is usually assumed to be zero and left out for simplification.

$$\tilde{\mathbf{m}} \simeq \mathbf{P}\tilde{\mathbf{M}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{M}} \tag{2.1}$$

$$\text{where} \quad \mathbf{K} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

For the calibration of a single camera, the extrinsic parameters are assumed to be the identity $[\mathbf{R}|\mathbf{t}] = [\mathbf{I}|\mathbf{0}]$, where $\mathbf{I}$ is a 3×3 diagonal identity matrix. Therefore, the only parameters to be estimated are the intrinsic parameters. However, if the single camera needs to be calibrated and localised in an existing work space then it becomes necessary to determine the extrinsic parameters relative to some known fixed points. For the case of multiple cameras, all cameras need to be localised with respect to each other and only one camera is assumed to be at the origin of the world coordinate system.

2.3.2 Camera calibration algorithms

The most popular calibration algorithms fall into four categories – 3D calibration, using a calibration object that provides known points in 3D space; 2D calibration, using an object where all the points lie on a plane; 1D calibration, where the calibration object is a set of points along a line; and self-calibration, which is unstructured and relies on correspondences being extracted from multiple views. Much of the foundations of camera calibration have been long established, however, this does not mean it is a solved problem. As will be seen, all four categories of calibration discussed in this section have novel new research being actively pursued and interesting problems to be solved. This includes applying the well understood algorithms to new and modern domains, such as in the case of self-calibration; applying modern approaches to improve accuracy, such as in the case of 2D calibration; and refinement of the underlying algorithm for use with fewer restrictions, such as with 1D calibration.

Self-calibration without calibration objects

Although most calibration methods to be discussed in this section make use of calibration objects, calibration is possible without the use of any such object, referred to as self-calibration or auto-calibration. A drawback of this is that the scale of the scene cannot be determined; a scene that is twice the size but also twice as far away would appear exactly the same. However, a benefit of this is that it can be applied to image sequences that were not taken with calibration in mind and therefore lack anything that can be used as a calibration object. The basic case for camera self-calibration requires at least $N \geq 7$ point correspondences across $M \geq 3$ images, between which the camera has undergone general motion. An important concept used here is the *image of the absolute conic* (IAC), $\omega = \mathbf{K}^{-T}\mathbf{K}^{-1}$, and its inverse the *dual image of the absolute conic* (DIAC), ω^*. These are point and line conic representations, respectively, of the projection into a given image plane of the complex conic located on the plane at infinity. Whilst not an actual view-able concept, self-calibration methods use constraints from epipolar geometry to compute and decompose these values into the intrinsic calibration matrix. This geometry is shown graphically in Figure 2.3. Point correspondences allow the fundamental matrix $\mathbf{F}$ and epipoles, $\mathbf{e}$ and $\mathbf{e}'$, for each image pair to be determined, which can be accurately estimated through well know approaches such as from Hartley [19, 45]. These are then used to apply constraints on the DIAC with what is referred to as the Kruppa equations, given in Equation 2.3, which relate the epipoles and fundamental matrix via the DIAC. Once the DIAC is estimated and decomposed into the intrinsic parameters, the extrinsic parameters can be estimated from the essential matrix using the method of Hartley [46].

$$[\mathbf{e}']_\times \mathbf{K}\mathbf{K}^T [\mathbf{e}']_\times^T \simeq \mathbf{F}\mathbf{K}\mathbf{K}^T\mathbf{F}^T \tag{2.3}$$

There are a variety of ways to solve the self-calibration problem — all of which are closely related. The first comes from the work of Faugeras et al. [16] which uses

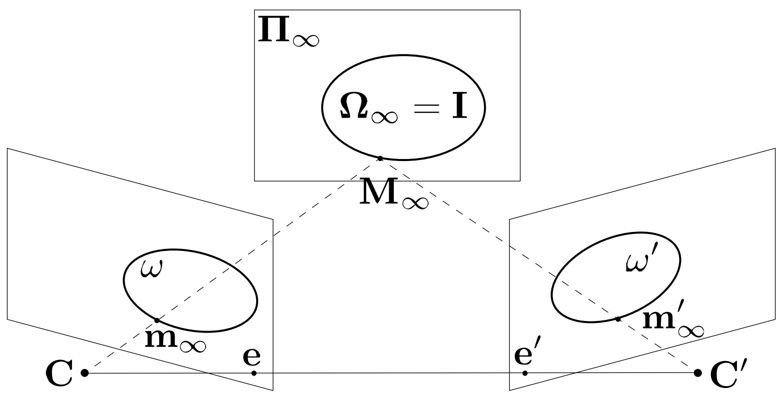


Figure 2.3: Geometry of self-calibration. $\mathbf{\Omega}_\infty$ is the *absolute conic* which is located on the plane at infinity, $\mathbf{\Pi}_\infty$. Its projection onto a camera with centre $\mathbf{C}$ is ω (respectively with $\mathbf{C}'$ and ω'). $\mathbf{M}_\infty$ is a point at infinity on the absolute conic and projects to $\mathbf{m}_\infty$ and $\mathbf{m}'_\infty$. $\mathbf{e}$ and $\mathbf{e}'$ are the epipoles on the left and right images respectively and correspond to the intersection between the line from $\mathbf{C}$ to $\mathbf{C}'$ and each image plane. If each camera has the same intrinsic parameters, then $\omega = \omega'$.

the Kruppa equations directly. Each image pair provides two linearly independent Kruppa equations for the five unknowns of the intrinsic parameters, and therefore three images are sufficient to solve the calibration. However, the Kruppa equations are polynomials of degree two and cannot be solved directly. Faugeras et al. used numerical continuation on five of the six equations to find the 32 possible solutions, then used the sixth to reject spurious estimations. Although their method is computationally expensive, they argue that it is preferable to nonlinear least squares due to the instability of the non-convex problem. This was followed up by the work of Luong and Faugeras [17] who expressed the polynomials of the Kruppa constrains based on an analysis of the properties of the essential matrix between two views. Overall, their method is very similar to the former, using the continuation method to find all 32 solutions to five equations. They improve the robustness by finding all solutions to all six combinations of five equations, then find the best from each list using a distance measurement, and finally find a single solution based on an average. Drawbacks of their method are that it is computationally expensive and only feasible with the minimal set of three images. It is also difficult to express constraints on the intrinsic parameters, such as the focal lengths being positive. Finally, their method is sensitive to noise.

There are alternative forms of the self-calibration problem that rely on different constraints. Triggs [47] derived a similar constraint to the Kruppa equations, instead based on the relationship between the DIAC and the absolute quadric, which is equivalent to the absolute conic but reduces the complexity of the problem. Their method allows additional constraints to be applied directly to the initial estimation, rather than relegating such constraints to a final bundle adjustment. Pollefeys and Van Gool [18] suggest a stratified approach consisting of three steps. Firstly, a projective reconstruction is estimated based on the infinite homography, that is the homography between image planes for points on the plane at infinity, which can be estimated from the fundamental matrix. Next, the projective calibration can be upgraded to an affine one by identifying the plane at infinity, and finally a metric calibration can be achieved from the affine calibration by using linear constraints between the infinite homography and the DIAC. Although the first and third steps are simpler than alternative methods, the second step relies on nonlinear constraints for the plane at infinity, which introduces similar issues to the nonlinear steps in the other algorithms discussed.

There are also a range of self-calibration algorithms that are based on specific motions of the camera or specific structures of the scene. Hartley [48] presented a method for cameras undergoing pure rotation which allows for constraints applied to the infinite homography. Armstrong et al. [49] derived a method that can determine metric calibration up to a two-fold ambiguity in scenes where the camera undergoes planar motion. Their method took advantage of constraints on the trifocal tensor, which is an image triplet analogue of the fundamental matrix. Both of these methods are interesting due to the fact that they rely on motion sequences that are degenerate for the Kruppa equation-based methods [50, 51]. The critical motion sequence that is inherent to self-calibration is pure translation, which is avoided in the method of Armstrong et al. [49] by allowing on-plane rotation, however, pure rotation and motion on a sphere are additional critical motion sequences specifically for the Kruppa equations [51].

One of the most straight-forward approaches to self-calibration can be seen in the multi-stage method of Gao and Hadha [52] which was later refined by Köhler [53]. The method is summarised in Figure 2.4 and first estimates the focal lengths under the assumption that they are the same in each direction and the principal point is at the centre of the image, then estimates the actual ratio of focal lengths, thirdly estimates the actual principal point, fourthly refines the focal lengths individually, and finally refines all parameters together. Each stage of the algorithm uses the results of the earlier state as a starting point. The initial assumptions of unity aspect ratio and principal point at the centre of the image are usually close enough to the true value that they do not result in excessive error. This multi-stage approach can be applied to any of the constraints mentions so far, however, both Gao and Hadha as well as Köhler rely on nonlinear least squares with a cost function enforcing the *equal singular value* constraint of the essential matrix. This method greatly improves

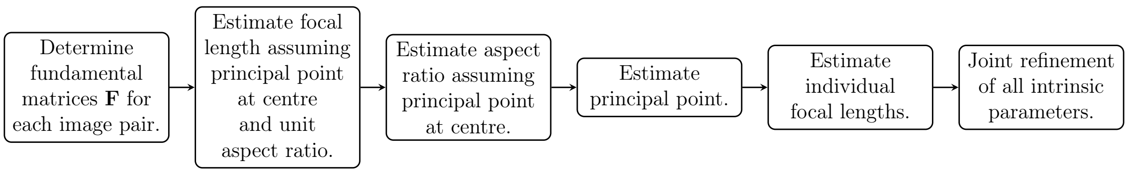
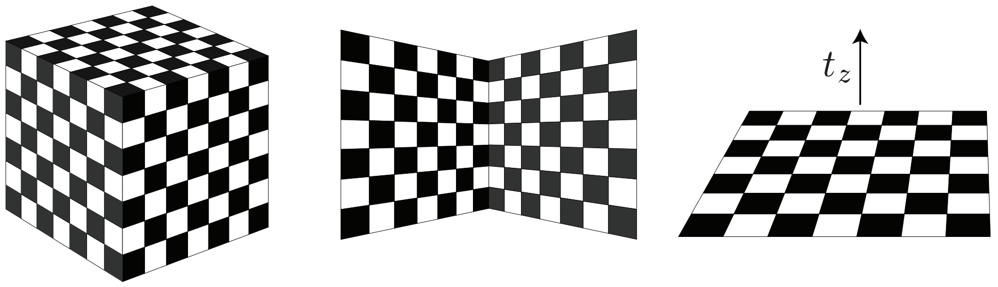

Determine fundamental matrices **F** for each image pair.
Estimate focal length assuming principal point at centre and unit aspect ratio.
Estimate aspect ratio assuming principal point at centre.
Estimate principal point.
Estimate individual focal lengths.
Joint refinement of all intrinsic parameters.

Figure 2.4: The multi-stage self-calibration process from point correspondences across at least three images, to joint refinement of intrinsic parameters. Each stage replaces some of the prior assumptions with the output of the previous stage.

the stability of the nonlinear estimation by limiting what is refined at a given time and improving initial estimates for each refinement, however, a drawback is that it requires repeated application of nonlinear refinement which can be computationally expensive.

Although much of the foundational work on self-calibration was established in the 1990s and early 2000s, the field is still highly active with attention being given to domain specific issues that introduce further complications or degenerate motions. For example, in many situations distortion of the camera, the modelling of which will be discussed further in the following section, can cause ambiguities in the calibration. The predominantly forward motion seen in SLAM applications results in an ambiguity as to whether the disparity between matched points is due to depth or radial distortion. Zhuang et al. [54] proposed a solution to this issue based on deep learning with a ResNet-34 architecture. Although actual SLAM footage doesn't have perfectly pure forward motion, their tests demonstrate that their learning approach estimated radial distortions much closer to the ground truth than the geometric method. Similarly, pan-tilt-zoom camera systems can have significant amounts of radial distortion that complicates calibration. Zhang et al. [55] also applied a deep learning approach to improve the calibration of such cameras with significant distortion and changing focal lengths, although, their approach relies on the assumptions of unity aspect ratio and principal point at the centre of the image. Tang et al. [56] applied self-calibration to the domain of pedestrian tracking where the pedestrians are modelled as poles of constant height moving across a flat plane. Using constraints on this model, they were able to identify the ground plane, horizon line, and vanishing points, which could be used to perform self-calibration. They used an evolutionary optimisation procedure to minimise reprojection error on points on the ground plane as well as optimising the distortion parameters to minimise variance in the heights of pedestrians over time. Overall, there is an interesting trend of solving issues relating to distortion and domain-specific camera or motion types, although current solutions still frequently rely on many assumptions to solve their problems.



(a) A calibration object of three perpendicular planes. **(b)** A calibration object of two perpendicular planes. **(c)** A calibration object of a single plane that undergoes a known translation.

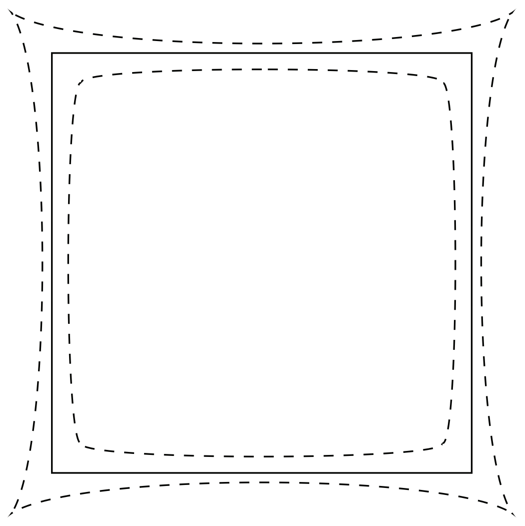
Figure 2.5: Types of calibration objects used in 3D calibration. Two or three perpendicular planes are often used, as in (a) and (b) respectively. A single plane can be used if photographed at two different locations separated by a known translation, as in (c).

Three-dimensional calibration objects

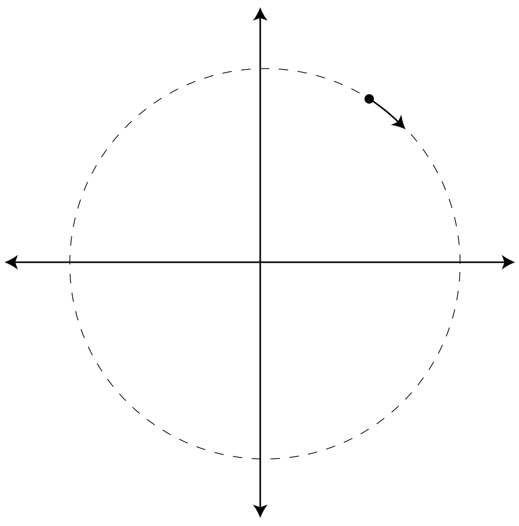
Much research has been done between the 1970s and 1990s in the intersection of the fields of close-range photogrammetry and computer vision in producing high-quality calibration for camera models that include both the aforementioned pinhole model as well as a range of distortions that can be present due to the geometry of the camera lens. Early work on this, which provides extremely accurate estimations, uses a 3D calibration object; that is, an object whose geometry provides a set of known points in 3D space. The calibration object is often in the form of two or three orthogonal planar checker-board patterns, such as in the work of Brown [10] or Heikkila and Silvén [11], or a single planar checker-board pattern that undergoes a known displacement between subsequent images as in the work of Tsai [12]. Examples of these types of calibration objects are represented in Figure 2.5, where a variable amount of planar checker-board patterns provide the known 3D points through edge detection.

In either of these scenarios the set of know 3D points can be extracted by performing corner detection on the images and are used to build a set of linear constraints according to Equation 2.1 on the vector $\mathbf{p}$ which contains the elements of the projection matrix $\mathbf{P}$. This produces a pair of constraints given in Equation 2.4 that can be stacked for each image points, solved, and decomposed according to the direct linear transformation method of Abdel-Aziz and Karara [57] to extract the intrinsic parameters as well as the extrinsic parameters relative to the coordinate frame defined by the calibration object.

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & x_iX_i & x_iY_i & x_iZ_i & x_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & y_iX_i & y_iY_i & y_iZ_i & y_i \end{bmatrix} \mathbf{p} = \mathbf{0} \tag{2.4}$$



(a) The effects of radial distortion. The solid line is the undistorted points and the dashed lines are distorted radially in a positive or negative direction.



(b) The effects of tangential distortion. The point is rotated about the distorted centre, along the dashed line.

Figure 2.6: Distortion models are usually reduced down to a combination of two types – radial and tangential. Radial distorts towards or away from the distortion centre based on distance from the centre, and tangential distorts about the distortion centre.

The 3D calibration problem begins with this linear step performed on the basic non-distorted pinhole camera model and is then usually followed by a nonlinear refinement to minimise the reprojection error between the observed corners and their expected projections based on the known 3D point locations according to a nonlinear model that includes various types of distortion. A range of distortion models have been considered in literature, which generally reduce down to radial and tangential components which are both modelled by power series. These different types of distortion include:

- **Radial distortion** which distorts the ideal image points away from or towards the distortion centre due to imperfect radial curvature of the lens.
- **Decentring distortion** which causes both radial and tangential distortion due to misalignment of the lens with the image sensor.
- **Thin prism distortion** which also causes both radial and tangential distortion due to imperfections in lens design or assembly, such as a slight tilt between lens elements.

In the literature, these different causes of distortion are reduced to two power series for the combined radial and tangential distortion effects which are demonstrated graphically in Figure 2.6. The radial component can be seen in models presented by Brown [10], Tsai [12], Weng et al. [58], and Heikkila and Silvén [11].

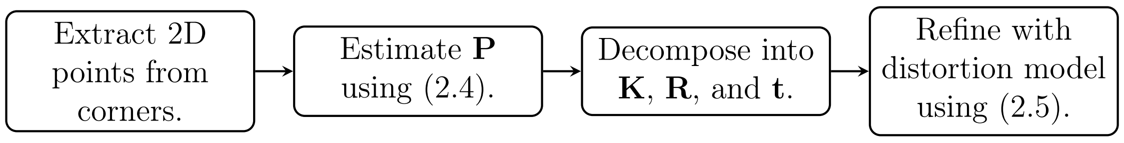

Extract 2D points from corners.
Estimate **P** using (2.4).
Decompose into **K**, **R**, and **t**.
Refine with distortion model using (2.5).

Figure 2.7: The 3D calibration process from input image(s) of checker-board patterns, to refinement of intrinsic, extrinsic, and distortion parameters.

Tsai omits the tangential component due to its insignificance compared to the radial component. Although Brown, Weng et al., and Heikkila and Silvén include tangential components in their distortion models, they all state that its effect is very small; Heikkila and Silvén found that tangential distortion was five to seven times smaller than radial distortion. The parameters to the power series modelling these two types of distortion are found using nonlinear least squares which optimises jointly for the pinhole model parameters and distortion parameters. This is most often done for the two most significant terms in each series. Representing these four terms in total by the four-vector ρ, this optimisation problem is written as in Equation 2.5.

$$\{\mathbf{K}^*, \mathbf{R}^*, \mathbf{t}^*, \rho^*\} = \underset{\{\mathbf{K},\mathbf{R},\mathbf{t},\rho\}}{\arg\min} \sum_{i=0}^{N-1} \frac{1}{2} ||\mathbf{m}_i - \pi(\mathbf{M}_i; \mathbf{K}, \mathbf{R}, \mathbf{t}, \rho)||_2^2 \tag{2.5}$$

From this, the full 3D calibration procedure can be summarised as a four-step process, given in Figure 2.7. Firstly, the image points corresponding to the known 3D points are extracted from the image (or images in the case of the displaced planar object) by means of edge detection. Secondly, the linear system of Equation 2.4 is solved using these points to determine the projection matrix **P**. Thirdly, the projection matrix is decomposed into the intrinsic parameter matrix **K** and extrinsic parameters **R** and **t**. Then finally, the pinhole model parameters are refined and the distortion parameters are estimated using the nonlinear refinement process of Equation 2.5.

Although these methods provide extremely high-quality calibration results, as is needed in close-range photogrammetry, there are drawbacks that limit their use in modern computer vision and robotic vision. The high quality calibration object is often expensive and difficult to use in comparison to the newer methods that will be discussed shortly, and many modern applications do not need the accuracy provided by them. A part of their difficulty in use stems from the smaller range of viewing angles that the calibration objects support; that is, the object becomes self-occluded at wider angles. As such, the 2D calibration method discussed next is the far more popular method in computer vision.

Two-dimensional calibration objects

Two-dimensional calibration is based on the work of Zhang [59] as well as Sturm and Maybank [60], and is one of the most popular and widely used forms of calibration due to its ease of use and incorporation into popular programming libraries and software platforms such as OpenCV [61] and Matlab [62]. Here, the calibration object is a single planar checker-board pattern that is observed at a number of arbitrary orientations. The method works based on the fact that it is equally valid to interpret the images of the planar object at arbitrary poses as if the object is stationary and instead the camera is moving. Therefore, they use the constraint that the planar object is on the XY-plane at $Z = 0$. This allows the third column of the rotation matrix to be removed, rewriting Equation 2.2 as the reduced form given in Equation 2.6.

$$\tilde{\mathbf{m}} \simeq \mathbf{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad \text{where } \mathbf{H} = \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \simeq \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} \tag{2.6}$$

Similar to 3D calibration, the observations build a linear system of equations which in this case is solved for the IAC that is then decomposed into the intrinsic parameters. In this case, the linear system of equations is build based on the constraints given in Equation 2.7, which are due to the fact that the columns of the rotation matrix are orthonormal.

$$\begin{aligned} \mathbf{h}_1 \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 &= 0 \\ \mathbf{h}_1 \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_1 &= \mathbf{h}_2 \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 \end{aligned} \tag{2.7}$$

Once the intrinsic parameters are known, the homography for each view can be fully determined and used to find the extrinsic parameters for each observation. Again, as with 3D calibration, distortion can be determined from a separate linear system of equations followed by a joint nonlinear refinement of all estimated parameters. In 2D calibration, the nonlinear least squares refinement computes the reprojection error over all of the N planar points across M views, as given in Equation 2.8.

$$\{\mathbf{K}^*, \rho^*, \mathbf{R}_j^*, \mathbf{t}_j^*\} = \underset{\{\mathbf{K}, \rho, \mathbf{R}_j, \mathbf{t}_j\}}{\arg\min} \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} \frac{1}{2} ||\mathbf{m}_{ij} - \pi(\mathbf{M}_i; \mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \rho)||_2^2 \tag{2.8}$$

The 2D calibration procedure can be summarised by Figure 2.8. Firstly, a checkerboard pattern is attached to a planar surface and imaged at a few arbitrary orientations, then the homography of each view is calculated. Each homography is used

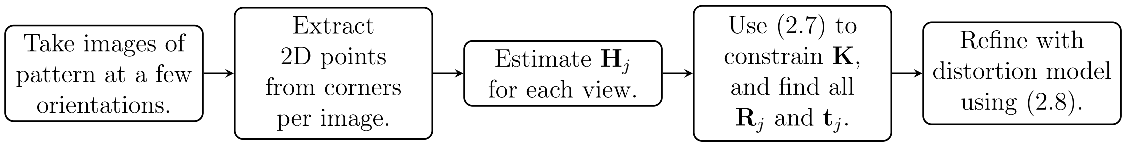
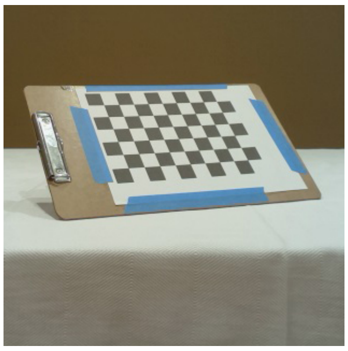
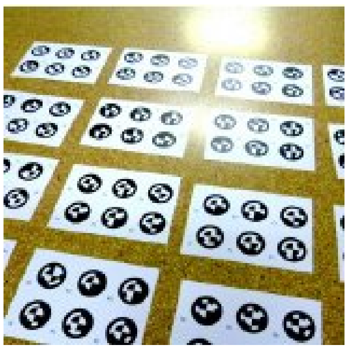

Take images of pattern at a few orientations.
Extract 2D points from corners per image.
Estimate $\mathbf{H}_j$ for each view.
Use (2.7) to constrain $\mathbf{K}$, and find all $\mathbf{R}_j$ and $\mathbf{t}_j$.
Refine with distortion model using (2.8).

Figure 2.8: The 2D calibration process from input image(s) of planar checker-board patterns, to refinement of intrinsic, extrinsic, and distortion parameters.


(a) A checker-board calibration plane.


(b) A fiducial marker calibration plane [63].

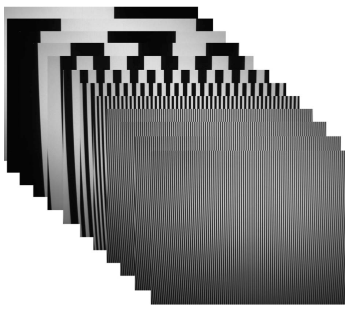

(c) A series of coded patterns for an active calibration plane (digital display) [64].

Figure 2.9: Types of planar objects used in 2D calibration. The basic algorithm uses the checker-board pattern seen in (a), however, fiducial markers (b) or active calibration (c) targets allow for image clipping and partial occlusion.

as constraints on the IAC which is decomposed to $\mathbf{K}$ which allows $\mathbf{R}_j$ and $\mathbf{t}_j$ to be determined from $\mathbf{H}_j$. Finally, a similar nonlinear refinement as in the 3D case is performed with the inclusion of a distortion model.

One of the great strengths of this algorithm is its ease-of-use. Computer vision applications that require metric calibration generally favour 2D calibration as it simply requires a checker-board pattern to be printed out and attached to a flat surface such as a clip board. Then, using freely available software, the calibration parameters of a camera are able to be quickly determined in a mostly automated process. There has been some work in improving the usability and accuracy of 2D calibration even further through the use of fiducial markers or active calibration targets. Fiducial markers are artificial landmarks that aim to facilitate high-quality and automatic point localisation, even with image clipping and partial occlusion. Active targets, on the other hand, are where digital displays are used as the planar object. Depictions of these two different types of planar pattern in contrast to the standard checker-board pattern can be seen in Figure 2.9.

One popular form of fiducial marker is ARTag which is a square pattern comprised of 8×8 bi-tonal cells where a two cell border of solid colour (black or white) facilitates blob detection and the inner 4×4 block encodes the unique ID of a given marker [65]. These markers have been specifically designed for fast detection and homography calculation with low false positive, false negative, and inter-marker confusion rates, as well as resistances to issues with illumination change, partial occlusion, and perspective changes. Fiducial markers have been utilised in 2D calibration in work such as that done by Fiala and Shu [66], where the markers are placed on a plane much like the the checker-board object. In their work, each marker provides four points towards the calibration problem. A similar approach was taken by Atcheson et al. [67] who used markers much like ARTag that were printed in alternating background colours, creating essentially a checker-board pattern with embedded fiducial markers. A similar approach was taken by Daftry et al. [63] who used circular fiducial markers laid out in a regular grid on a planar base. The benefits of using fiducial markers are that individual points in the pattern can be localised even in cases where the pattern is clipped at the image border or partially occluded by objects in the scene. The former benefit is particularly useful in that it allows calibration points to be taken close to the image border and corners where distortion is the greatest.

An example of using active targets in 2D calibration is seen in the work of Schmalz et al. [64] where the authors use a digital display as the planar object. They propose to fill the entire field-of-view of the camera with the screen and then use a series of coded patterns to uniquely identify each pixel. Similar to the case with fiducial markers, this method allows points near the image corners to be utilised to more accurately estimate the distortion parameters. Their method sees a five-fold reduction in reprojection error compared to using a standard checker-board pattern. Although they discuss the similar benefits between active targets and fiducial markers, they do not directly compare the accuracy of the two classes of methods. They argue that their method has comparable ease-of-use as the basic checker-board pattern due to the ubiquity of digital displays, however, their method assumes that a camera can be positioned to have its field-of-view filled. Comparatively, the methods using fiducial markers have the exact same ease-of-use as the checker-board pattern with the added benefit of automatic image acquisition whenever a defined target is in view.

One-dimensional calibration objects

Calibration with a one-dimensional object is another method also proposed by Zhang [14] shortly after his work on two-dimensional calibration, however, this form has not seen as wide adoption. 1D calibration involves an object of at least $N \geq 3$ collinear points with known distances where one point is fixed, $\mathbf{A}$, about which the other two move, $\mathbf{B}$ and $\mathbf{C}$, shown in Figure 2.10. The point $\mathbf{C}$ is the midpoint, with the ratios between overall length and distance to either end given as λ_A and λ_B, respectively. The corresponding image points are $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, respectively. This method uses constraints on the cross-products between the points on the line, giving the

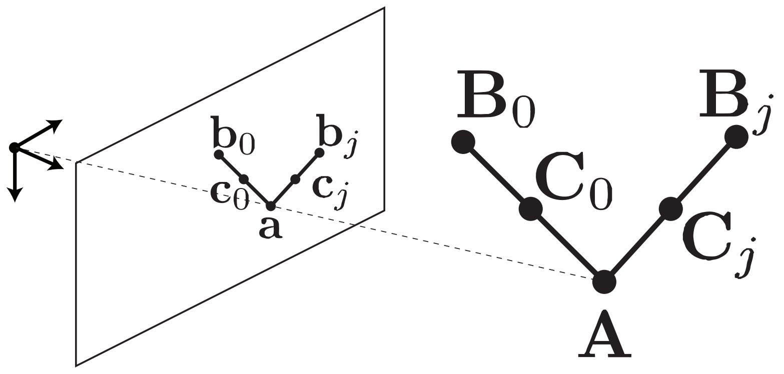


Figure 2.10: One dimensional calibration, using an object comprised of three collinear points, shown at two orientations. The rotation between the two orientations occurs about a fixed point.

relative depth β in Equation 2.9, to build a linear system of equations that can be solved for the IAC based on the constraint in Equation 2.10, where z_A is the unknown depth of the fixed point. The problem has 5 unknowns for the intrinsic parameters, 3 more for the fixed point, and an additional 2 per image defining the line. In return, there are 2 constraints provided by the observation of the fixed point and a further 3 per image based on observations of the remaining points. Therefore, this method requires at least $M \geq 6$ observations in order to solve for the calibration.

$$\beta = \frac{\lambda_A(\tilde{\mathbf{a}} \times \tilde{\mathbf{c}}) \cdot (\tilde{\mathbf{b}} \times \tilde{\mathbf{c}})}{\lambda_B(\tilde{\mathbf{b}} \times \tilde{\mathbf{c}}) \cdot (\tilde{\mathbf{b}} \times \tilde{\mathbf{c}})} \tag{2.9}$$

$$z_A^2(\tilde{\mathbf{a}} + \beta\tilde{\mathbf{b}})^T\mathbf{K}^{-T}\mathbf{K}^{-1}(\tilde{\mathbf{a}} + \beta\tilde{\mathbf{b}}) = L^2 \tag{2.10}$$

As with 3D and 2D calibration, this process is usually followed by a separate linear estimation of distortion parameters and then jointly refined with nonlinear optimisation. This nonlinear least squares problem optimises the reprojection error of the $N \geq 3$ collinear points across $M \geq 6$ views, refining the intrinsic parameters, distortion parameters, as well as the estimated 3D positions of the calibration object points in each observation. The objective function is given in Equation 2.11. The full procedure is given in Figure 2.11.

$$\begin{aligned}\{\mathbf{K}^*, \rho^*, \mathbf{A}^*, \mathbf{B}_j^*, \mathbf{C}_j^*\} = \underset{\{\mathbf{K},\rho,\mathbf{A},\mathbf{B}_j,\mathbf{C}_j\}}{\arg\min} \sum_{j=0}^{M-1} \frac{1}{2}(||\mathbf{a}_j - \pi(\mathbf{A}; \mathbf{K}, \rho)||_2^2 \\ + ||\mathbf{b}_j - \pi(\mathbf{B}_j; \mathbf{K}, \rho)||_2^2 + ||\mathbf{c}_j - \pi(\mathbf{C}_j; \mathbf{K}, \rho)||_2^2)\end{aligned} \tag{2.11}$$

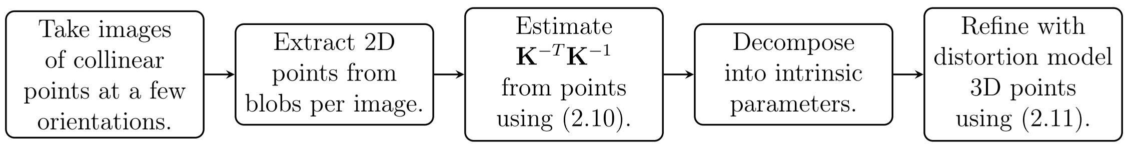

Take images of collinear points at a few orientations.
Extract 2D points from blobs per image.
Estimate $\mathbf{K}^{-T}\mathbf{K}^{-1}$ from points using (2.10).
Decompose into intrinsic parameters.
Refine with distortion model 3D points using (2.11).

Figure 2.11: The 1D calibration process from input images of collinear points, to refinement of intrinsic, extrinsic, and distortion parameters.

Zhang further extended the method to show it was possible to solve with only observations of the two moving points by calculating the location of the unseen fixed point. In fact, his work shows a small improvement in accuracy when estimating the fixed point rather than observing it, which he attributed to the elimination of human error in both annotating the fixed point as well as holding the fixed point in place. The critical motions of this algorithm have been analysed by Hammarstedt et al. [68] finding that calibration fails if the motion of the object lies on a cone; the authors suggest that a 'zig-zag' motion is ideal for ensuring successful calibration.

The base algorithm for 1D calibration has stricter constraints on motion than for 2D and 3D calibration which work against its adoption over that of the far more popular 2D. However, there has been some work in improving the base algorithm and relaxing these constraints. Wu et al. [69] analysed the allowable motion patterns for successful calibration from a geometric standpoint and found that the 'fixed' point was, in fact, able to move as long as that motion was constrained to a plane. They achieved this in practice by observing the calibration pattern at various locations and orientations whilst lying on a flat surface. This approach, however, eliminates some benefits that 1D has over 2D; the 1D object normally doesn't self-occlude at wide angles like the 2D pattern, but if it is resting on a fixed planar surface then it has the same occlusion properties as the 2D case. Qi et al. [70, 71] suggest that the most feasible manner to realise this relaxation is by tossing the object in the air, allowing it to move in parabolic motion under gravity. They found that only the 'fixed' point needs to be constrained to the plane, and if it is the centre of gravity, then it will be constrained to the planar surface of the parabolic motion. Although this form of motion is far simpler in practice compared to that of Wu et al., the authors found that it had about five times the error comparatively in estimations on synthetic data. A reason for this is that these geometric methods ultimately rely on the estimation of vanishing points, which are highly susceptible to noise, and the method of Qi et al. has a greater likelihood of small angles of rotation between pattern observations which negatively affects the vanishing point estimations.

There has also been some work on improving the numerical stability and accuracy of Zhang's base algorithm. De França et al. [72] utilised the normalisation technique of Hartley [45] to reduce the effects of noise, then improved the speed of the non-linear refinement stage by adopting a partitioned Levenberg-Marquardt optimisation. Their approach improved both the accuracy and speed of the algorithm, however, they only demonstrated its application to Zhang's original algorithm and not to the geometric versions with relaxed fixed-point constraints of Wu et al. [69] and Qi et al. [71]. Shi et al. [73] and Wang et al. [74] both also focused on improving Zhang's form of the algorithm with different methods of estimating the relative depth. Shi et al. formulated an alternative similarity-invariant calculation of the relative depth, which they argue does not need nor benefit from the normalisation of de França et al. Wang et al., on the other hand, applied the normalisation directly to the estimation of the relative depth, rather than to the overall measurement matrix. Both authors then used a robust error model from the relative depth estimation to weight the linear algorithm for further improved accuracy. These improvements mean that Zhang's form of the algorithm sees more accurate calibration than the variants with relaxed constraints on the fixed-point, resulting in a choice between ease-of-use and accuracy. However, similar improvements could be explored for the formulation of Qi et al. to reduce disparity between the methods.

2.3.3 Multi-camera calibration and localisation

Multi-camera calibration involves determining the intrinsic parameters for a number of different cameras simultaneously, whilst also estimating a globally consistent localisation. The localisation is of particular importance in multi-camera systems because most computer vision tasks cannot take advantage of multiple views without knowing the relative transformation between said views. Each of the types of calibration discussed in the single camera case can be applied to multi-camera systems with different strengths and advantages as well as additional considerations to whether all of the cameras in the network have overlapping or non-overlapping fields-of-view (FOV).

The usage of self-calibration across multiple views is the most common choice in the general case where scene scale is not required due to the straight forward nature of the application. The general form of the Kruppa equations does not require that each image be associated with the same intrinsic parameters. Therefore, rather than having a single moving camera, this method allows for a number of stationary cameras with varying intrinsics taking a single view each can be used to achieve calibration. Pollefeys et al. [75] showed that the minimal constraint for successful self-calibration with varying parameters is that the skew is zero, which is a safe assumption in modern cameras. They demonstrated successful multi-camera self-calibration from at least eight views using the absolute quadric method first suggested by Triggs [47]. By adopting further assumptions of unity aspect ratio and the principal point being at the centre of the image they were able to find an initial linear estimate with only two views which could later be refined with nonlinear optimisation. Lourakis and Deriche [76] found similar results using the Kruppa equations. They also were able to perform multi-camera self-calibration with at least eight stationary views, or with only two views given the same assumptions for an initial estimate that was then refined. These methods are fairly straightforward, achieve good accuracy, and are simple to use in multi-camera systems due to no issues with occlusion of a calibration object. However, as mentioned they are unable to determine the scale of the scene and therefore any extrinsic localisation is only determined up to a similarity.

More recent work on multi-camera self calibration has explored situations with more limiting constraints. Brückner and Denzler [77] outlined a method aimed at pan-tilt-zoom cameras with limited correspondences where they use a probabilistic method to determine FOV overlap and actively rotate the cameras to achieve better pairwise overlap. This also greatly improves the consistency of a global scale estimate. Whilst this scale is not the true scale of the scene, it is crucial for it to be consistent between views for subsequent computer vision applications to be successful. Sun and Xu [78] also explored multi-camera self-calibration with limited FOV overlap. Instead of relying on feature point correspondences between views, they track correspondences of a moving object across multiple images from each view separately. The intrinsic self-calibration is performed locally and then the camera is localised with regard to a coordinate frame attached to the moving object. The object's visibility in the different views allows a globally consistent localisation to be achieved. Continuing the trend of self-calibration with limited correspondences, Vasconcelos et al. [79] considered the case where an uncalibrated camera is being added to an existing calibrated and localised network. The standard method would rely on triple correspondences where a feature point seen by the new camera is matched with points in at least two other views. Their method instead used separate pairwise correspondences against each of the other views, which was a much more practical requirement in scenes with limited overlap.

Multi-camera calibration utilising 3D or 2D calibration objects is less popular than self-calibration due to the fact that the calibration objects can self-occlude quite easily at wide angles. However, there has been some novel work at limiting the impact of this issue. Shen et al. [80] produced an interesting approach to 3D calibration that utilised a world globe as the calibration object. Mathematically, their approach is more similar to Zhang's 1D calibration [14] as they use points on the surface of the globe to estimate the centre and utilise constraints on estimated poles through the globe. The use of the globe rather than the perpendicular planes of standard 3D calibration means that the object can be easily viewed from all directions. 2D calibration has also seen adaption to multi-camera systems with attention to non-overlapping FOVs and self-occlusion issues. Liu et al. [81] considered a system of two cameras without overlapping FOVs and designed a flexible calibration target comprised of two planar checkerboards separated by a known and constant rigid tansformation. This allowed each camera to view one of the sub-targets which were each used for standard 2D calibration. This was extended by Xia et al. [82] who designed the multi-checkerboard object to be re-configurable for more flexible arrangement of the non-overlapping cameras and then used a method using a single calibrated camera and fiducial markers to accurately determine the rigid transformation between the sub-targets of a given configuration. Zhao et al. [83] also considered non-overlapping FOVs in designing a method where separate cameras were calibrated using 2D calibration individually, but each camera had a planar fiducial marker attached to it that could be viewed by an additional calibrated support camera which was used to determine the global localisation. These methods resolve issues with occlusion and non-overlapping FOVs, however, their reliance on additional calibrated cameras limits their use. Feng et al. [84] considered just the issue of self-occlusion of the calibration object in multi-camera systems and designed a planar calibration object which was a flat pane of glass with the checkerboard pattern printed on one side. This allowed the object to be viewed from both sides by an array of cameras. One issue that they needed to solve was that the cameras on the back side of the object were viewing the pattern under refraction through the glass which would lead to less accurate calibration. They solved this by modelling this refraction in their calibration problem for the affected cameras. There is still the problem of self-occlusion for cameras at wide angles with respect to the calibration object, however, as the object is moved and rotated arbitrarily this would only affect some of the cameras at a time and not all images seen by a given camera.

Calibration with one-dimensional object is also appealing for multi-camera calibration due to fewer issues with self-occlusion compared to the case of viewing the 2D calibration pattern from wide angles. It also has advantages over self-calibration as the calibration object is able to provide a known scene scale, producing metric localisation. A drawback to 1D calibration that can dissuade its use is the constraint of the fixed-point, which can be difficult to achieve in practice. However, this constraint can be eliminated entirely in multi-camera calibration setups. The first instance of multi-camera calibration and localisation using a free-moving 1D object was presented by Kojima et al. [85], however, their method required a 'reference' camera with known intrinsic parameters. Their approach computes the infinite homography between the reference camera and a target camera based on vanishing points of the 1D object, and then uses knowledge of the intrinsic parameters of the reference camera to determine the intrinsic parameters of the target camera as well as its orientation. Finally, constraints on the 1D object are used to determine the the position of the target camera. Although this algorithm allows for free motion of the calibration object, the need for a calibrated base camera limits the scenarios in which it can be used and reliance on computation of vanishing points is highly susceptible to noise as has been mentioned with regard to other algorithms. This work was followed up by Wang et al. [86] who removed the need for a calibrated reference camera. Their method also computes the infinite homography from vanishing points, but instead of using known parameters to decompose the infinite homography they instead use it to compute an affine projection matrix which provides an 'affine' calibration. This affine calibration is then 'upgraded' to Euclidean metric calibration using constraints on the 1D object. The remaining issue of relying on vanishing points was later remedied in work by de França et al. [87] by instead calculating the infinite homography from the fundamental matrix and epipoles. This method has much greater stability in the presence of noise than methods using vanishing points. A remaining issue in the approach of de França et al. is that they perform two separate stages of non-linear refinement which can slow down the algorithm, and the first of which relies on constraints that are not directly geometrically meaningful.

Self-calibration and 1D calibration adapt the best to multi-camera systems with overlapping fields of view. Although there have been a number of methods discussed for 2D calibration in such systems, they are not as flexible. Self-calibration only requires enough of an overlap for point correspondences, however, it does require eight cameras if only zero skew is assumed or two cameras if assumptions are also made for aspect ratio and principal point. Despite this, those assumptions are usually close enough to the true values that bundle adjustment will converge correctly. In multi-camera systems, the only requirement for 1D calibration is that the cameras can all view the calibration object for at least the minimal number of observations. 2D calibration, on the other hand, still exhibits a degree of self-occlusion at wide angles even with the improvements discussed above. Overall, if the metric scale of the scene is not required then self-calibration is an excellent choice for multi-camera systems, and if metric scale is needed for the application then 1D calibration is necessary and preferable to the other methods using calibration objects.

2.3.4 Summary and open problems

The key findings of this section relate to the types of calibration and their applicability to multi-camera systems. The four classes of algorithms discussed are 3D, seen in the work of Brown [10] and Tsai [12]; 2D, from the work of Zhang [59]; 1D, also from Zhang [14]; and finally self-calibration, which has been seen in many forms including the work of Faugeras et al. [16]. 3D calibration is the least common used in robotic vision due to its relative inflexibility. 2D calibration is definitely the most widely adopted for this field due to the simplicity of printing off a checkerboard pattern and affixing it to a flat object, as well as its inclusion in software and libraries such as Matlab and OpenCV. Self-calibration is also widely popular in robotic vision due to its ease of use. In scenarios with a mobile camera where the scene provides effective feature points, self-calibration is highly effective; however, a clear drawback is the lack of scene scale which can immediately disqualify it as an option in certain applications. The final class of methods discussed was 1D calibration, which is still a highly active area of research, although, does not yet have the wide adoption of 2D and self-calibration. Compared to 2D calibration, the creation of a 'calibration wand' is clearly not as trivial as the checkerboard pattern, however, once a suitable calibration object is procured the method does enjoy the same ease of use. Overall, for use in single-camera systems, 2D calibration and self-calibration stand ahead as the most common choices. This changes, however, in multi-camera systems — self-calibration is still a common choice due to ease-of-use, but 2D calibration loses favour due to self-occlusion of the calibration object at wide angles. Both 2D and 3D calibration operate best when observing keypoints predominantly from the front, whereas multi-camera systems can often have much wider camera angles. It is in this scenario where 1D calibration gains multiple strengths: firstly, the multi-camera system allows constraints on the motion of the calibration object to be relaxed, allowing for full general motion; and secondly, the nature of the calibration object removes any issues with self-occlusion, allowing cameras to observe the keypoints from all directions. It is for these reasons that the two best candidates for calibration in multi-camera systems are self-calibration and 1D calibration, however, multi-view 1D calibration is still an active area of research that does not yet have algorithms that are as straight-forward nor widely adopted. These open problems relating to the present work can be summarised as follows:

- Current one-dimensional calibration algorithms offer a choice between ease-of-use and accuracy; however, alternative formulations can be explored to reduce the disparity between the two methods.
- Novel types of calibration objects, such as cubes, globes, and double-sided planes, can alleviate self-occlusion issues.
- Multi-camera calibration algorithms, particularly one-dimensional ones, are still an active area of research that do not yet have as straight-forward approaches as the alternative algorithms with adoption in popular software platforms and programming libraries.

2.4 Visual odometry and visual SLAM

Visual odometry is the problem of tracking a robot's motion over a trajectory by estimating the relative poses between each image frame from a camera attached to the robot and building a map from these images and poses. Visual *simultaneous localisation and mapping* (visual-SLAM) is an extension of this problem where the long-term consistency of the map is also a priority of the system. This section explores some of the main contributions to the fields of visual odometry and SLAM and how these works relate to each other. This is done both from the point of view of major paradigms as well as through the lens of a general keyframe-based framework which is becoming the *de facto* standard. Section 2.4.1 discusses nineteen significant VO and SLAM implementations from the past 15 years by categorising them in four ways – whether they are labelled as VO or SLAM, how they fit into the *direct* vs *indirect* paradigms, whether they utilise their input images in a dense or sparse manner, and what types of input data they accept. Section 2.4.2 then explores these implementations in greater detail by applying them to a keyframe SLAM framework. This involves dividing the problem into three threads with different responsibilities and sub problems, then discussing how each algorithm solves a subset of these problems. From this, one can see that the algorithms that are labelled as visual odometry generally implement a smaller subset of the problems than those labelled as full SLAM systems. The motivation of this is to identify suitable paradigms, algorithms, and approaches for the second component of Figure 2.1. A summary of these findings and the related open problems is then given in Section 2.4.3.

2.4.1 VO and SLAM paradigms

Visual odometry and SLAM can generally be categorised in four main ways - whether the algorithm is VO or SLAM, direct versus indirect formulations, dense versus sparse image usage, and which types of cameras and inputs they support. As for labelling an algorithm VO or SLAM, this generally comes down to what range of features are implemented particularly relating to long-term success, with visual odometry implementing a subset of the features found in SLAM. Considering the next two categorisations, there are four possibilities - indirect and dense, direct and dense, indirect and sparse, or direct and sparse. The latter two are the more common methods seen in modern algorithms. Concerning the fourth categorisation, these are not mutually exclusive and it is common to see state-of-the-art systems being expanded to support as many input types as possible. The types of input data looked at here are monocular cameras, stereo camera systems, RGB-D colour and depth cameras, omni-directional or fisheye cameras, and IMU data. Additionally, we consider whether a system supports semantic labelling of the global map. A summary of this information can be seen in Table 2.1.

Visual odometry or SLAM?

As mentioned, visual odometry aims to estimate relative poses between images and SLAM aims to provide long-term consistency to the resulting map. Although there is no hard-and-fast definition for visual odometry and SLAM that draws a clear line where one ends and the other begins, there is a modern trend for different systems to more clearly fall into one category or the other. The general consensus is that visual odometry systems are more concerned with the local pose estimates frame-to-frame, or keyframe-to-keyframe. In some purely VO systems, the map is only a means-to-an-end for consistent local poses. Although VO systems might use keyframe-based designs and have local optimisation, often of both the poses and the map, they do not have systems in place to deal with long-term accumulation of drift. SLAM systems, on the other hand, use relocalisation features to gracefully handle tracking loss, loop detection and closure to eliminate drift when revisiting already mapped locations, expensive global optimisation to improve the consistency of all estimations, and map reuse to allow multi-session mapping.

It should be noted, however, that not all SLAM systems implement all of these features and not all VO systems omit them. In Table 2.1, each system has been labelled based on how the authors refer to their own work. This does mean that some of the older algorithms that are referred to as SLAM implement a similar feature-set to some of the more recent algorithms that are referred to as VO, revealing a trend in modern SLAM systems to rely on a wider array of long-term mapping features. This is shown in more detail in Section 2.4.2, where a general keyframe-based visual SLAM architecture is given in Figure 2.14 and is analysed with respect to each of the discussed algorithms, which is summarised in Tables 2.2-2.4 where blank spaces can be clearly seen for algorithms that do not implement the relevant features.

Earlier works such as MonoSLAM [88], Parallel Tracking and Mapping (PTAM) [20], Dense Tracking and Mapping (DTAM) [89], and KinectFusion [90] are all referred to by their authors as SLAM systems despite lacking features such as relocalisation and loop closure, and PTAM being the only algorithm amongst those four that implements a global optimisation in the form of bundle adjustment. Further contrast can be seen between MonoSLAM and the Multi-state Constraint Kalman Filter (MSCKF) of Mourikis and Roumeliotis [91], the latter of which is identified as a VO algorithm despite both algorithms being based on somewhat similar extended Kalman filters (EKF) designs with similar levels of features.

The contrast between VO and SLAM can also be seen between Large-scale Direct SLAM (LSD-SLAM) [92] and Direct Sparse Odometry (DSO) [93] which have very similar designs. Both are keyframe-based direct methods with the main change between them being that DSO has moved to a sparse formulation and included an affine lighting model. However, the difference that makes the former a SLAM system and the latter a VO system is that LSD-SLAM builds a graphical representation of the keyframes and uses OpenFABMAP [22] to detect loops which are added for pose-graph optimisation [94]. This loop closure feature extends the operating lifetime of the map by reducing drift and thereby elevates the algorithm to the classification of SLAM. Similar constrast can be seen in a comparison between KinectFusion and later algorithms such SLAM++ [95] and ElasticFusion [96]. All three algorithms are based on iterative closest point (ICP) optimisation of point clouds from RGB-D cameras, however, KinectFusion only has local tracking features with nothing to maintain the long term consistency on the global map. SLAM++ maintains the map by comparing the local portion with fragments of older portions as a form of loop closure, and ElasticFusion uses an appearance-based lookup to detect these loop closures. Therefore, by a modern standard one might classify KinectFusion as simply VO in comparison to these newer algorithms that can confidently be called SLAM.

From these comparisons it can be seen that it is the addition of some or all of relocalisation, loop closure and global optimisation features with the goal of producing a long-term globally consistent map that elevates a VO algorithm to SLAM.

Direct vs indirect

The distinction between direct and indirect methods relates to the underlying frame-to-frame or frame-to-keyframe tracking method, illustrated in Figure 2.12, and has significant ramifications on most facets of the VO or SLAM system. Indirect methods are the more traditional formulation and involve the extraction of an intermediate data-type which is then matched to form correspondences between images, whereas direct methods use and align the raw image data itself to determine the poses. These two formulations lead to fundamentally different pipeline details.

Indirect methods, also called feature-based methods or geometric methods, most commonly have three stages to tracking. Firstly, feature points are robustly detected and descriptors are extracted in each frame, then feature descriptors are matched between frames to form correspondences, and finally these correspondences are used in a geometric approach to determine the pose. Most indirect methods detect features using algorithms such as the Harris corner detector [97], the Shi-Tomasi corner detector [98], or the FAST feature detector [99]. Systems that make use of ORB features, like ORB-SLAM [100], generally make sure of the associated feature detector which is an orientated FAST detector [101]. Once the feature keypoints have been detected the descriptors need to be extracted. Many systems such as MonoSLAM [88], PTAM [20], and SVO [102] use simple image patches as feature descriptors, whereas systems such as ORB-SLAM [100] and parts of BASALT [103] use robust ORB descriptors which are more resistant to lighting, scale, and rotation changes between images [101]. These features are then matched either by guided search or nearest neighbour comparisons in order to determine correspondences and then used in a perspective-n-point RANSAC algorithm to determine the pose [104].

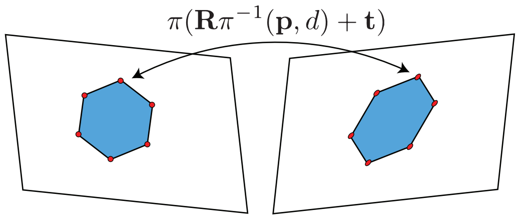
Direct methods, on the other hand, don't involve any intermediate representation of the image – they use the image intensities themselves. This can be seen in systems such as DTAM [89], LSD-SLAM [92], ROVIO [105], and DSO [93] where whole-image alignment is used to determine the pose beteen camera frames. This is generally done using the Lucas-Kanade algorithm which iteratively refines the pose required for a template image to register photometrically with another image using a Gauss-Newton gradient descent approach [106]. One limitation with the Lucas-Kanade approach is that depth values are required in one of the images in order to warp the pixels from one view to the other to compare intensity values. To determine these values, DTAM initialises using a standard feature-based approach, however, LSD-SLAM and DSO avoid the issues entirely by initialising with a random depth map high variance and allowing the system to 'snap' to a particular scale in the first few frames.

Not all methods fall neatly into these two paradigms, however. Semi-direct Visual Odometry (SVO) [102] is an example of an approach that has aspects of direct and indirect methods. In SVO, direct methods are used first to produce an initial estimate of the pose between the image frames, then indirect methods are used to extract features and match them to sub-pixel locations based on that initial estimate. This allows the system to determine feature correspondences which can be used in traditional bundle adjustment. Another group of SLAM systems that cannot cleanly be labelled with the above definitions include KinectFusion [90], SLAM++ [95], and ElasticFusion [96]. These systems are RGB-D SLAM algorithms that can be labelled indirect in that they involve an intermediate representation of the images as point clouds, however, their methods of aligning point clouds using Gauss-Newton ICP is more similar to the direct pipeline than the feature-matching indirect pipeline. In fact, ElasticFusion involves a joint refinement that aligns the point cloud geometrically and photometrically, and can be considered a hybrid approach like SVO – albeit a very different style hybrid approach.

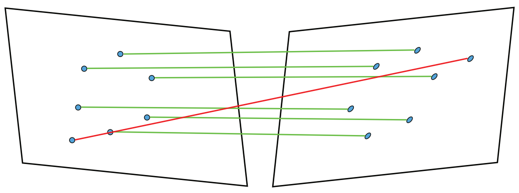
As can be seen in Table 2.1, most VO and SLAM systems can be considered either indirect or direct, although with some utilising aspects of both in a hybrid manner.

Dense vs sparse

The density of the VO or SLAM system refers to how much of each input image is used. Dense methods utilise every pixel of the image and rely on having the per-pixel



(a) Direct methods.



(b) Indirect methods.

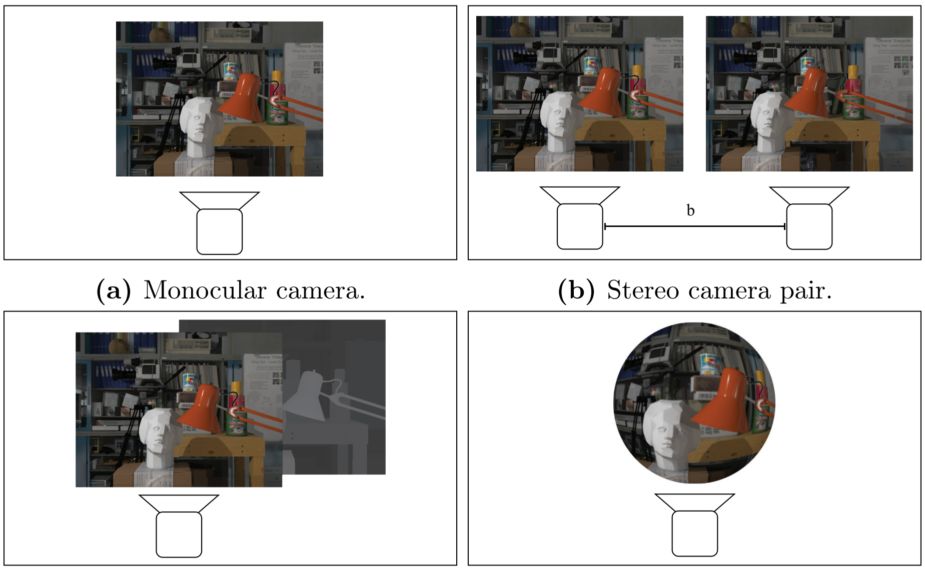
Figure 2.12: Direct methods relate points implicitly by a warp which is refined by minimising the error between raw pixel values. Sparse direct methods are indicated by the points of the shape, while dense direct methods are indicated by the entire shape. Indirect methods find correspondences through robust feature matching and use RANSAC to find the pose in the presence of outliers.

computational time fast enough to achieve real-time performance. Sparse methods, on the other hand, strive for real-time performance by only using the best selection of pixels.

Earlier forms of direct methods were predominantly dense. DTAM [89] is dense in both its image alignment and depth map estimation, which are both formulated as optimisation problems. LSD-SLAM [92] can be considered as *semi-dense* in that it uses all pixels that it is able to, however, it does reject many pixels during its depth map building stage. A key strength of these methods is that they are able to use information in all regions of the image regardless of the presence of textures or corners. This means that these dense and direct methods can be superior in certain scenes and environments where feature-based methods would struggle to find enough keypoints to match. RGB-D SLAM methods such as KinectFusion [90], SLAM++ [95], and ElasticFusion [96] are also dense in their formulation. They use their depth images to generate a point cloud with a vertex for every pixel, then determine poses by patching this point cloud to the global one.

Sparse methods are more common due to the huge computational savings afforded by only processing a small number of points in each image. This has been the case for most feature-based indirect methods, with early systems like MonoSLAM [88] and PTAM [20] both use very few features with the former having 12 features maintained in the map and the latter initially only relying on a 50 point search for correspondences. More modern systems such as ORB-SLAM [100] work on 1000-2000 corner points per image, but this is still hundreds to thousands times fewer than dense methods. Most modern direct methods have also moved towards using more sparse approaches per frame. DSO [93], for example, aims to register 2000 points between images which are chosen to be both well distributed across the image and have high gradients relative to their local region. SVO [102] also uses sparse approaches to both its direct and indirect components.

Although one of the arguments for dense direct methods was that sparse feature



(a) Monocular camera. **(b)** Stereo camera pair.

(c) RGB-D camera. **(d)** Omni-directional or fish-eye camera.

Figure 2.13: Camera types include monocular cameras which provide a single view, stereo camera pairs which provide two views separated by a known baseline, RGB-D cameras which provide a single colour image view and a depth map, and omni-directional camera which provide a single but very wide field of view.

selection based on corners would fail in low-texture environments, modern sparse direct methods generally still claim to have this advantage as their point selection is usually designed to sample from low-texture regions as well as high-texture regions to ensure that the points have good coverage across the image. For example, in DSO [93] the point selection scheme iteratively reduces the gradient threshold until it gets enough points with a good distribution. While dense direct methods use all high- and low- texture regions, sparse direct methods *can* utilise all regions where as feature-based sparse methods can only utilise high-texture regions.

Input types

The VO and SLAM systems looked at in this section have input type including monocular cameras, stereo cameras systems, RGB-D cameras, omnidirectional or fisheye cameras, and IMU sensors. The monocular camera is the most common camera type seen in the systems looked at here, with most systems that support multiple input types starting as monocular only and adding stereo, RGB-D, or IMU support later on. A comparison of these camera types is given in Figure 2.13

Monocular cameras are the most popular cameras seen in VO and SLAM, mainly due to their availability at high quality and low cost in commodity hardware as well as their ease-of-use. A monocular camera is inherently a bearing sensor, meaning that it cannot determine the distance to any objects from a single frame but instead only that the object lies on a line from the camera's optical centre. In order to gain a sense of depth, monocular systems require temporal-stereo match – that is, correspondences across frames at different times which should provide some amount of parallax. In feature-based monocular methods, such as in ORB-SLAM [100], 3D locations of points can be determined by triangulating the explicitly matched features once the pose has been estimated. In direct methods, such as LSD-SLAM [92] or DSO [93], correspondences are searched for along epipolar lines after estimating the pose which reveals the depth. Most monocular VO and SLAM methods need to alternate between estimating the pose and determining depths or 3D points.

Stereo cameras are able to determine depth much more easily. These systems use pairs of cameras that are separated by a known baseline, which allows for correspondences to be found by simply scanning along the images horizontally for a match and using the known baseline distance to infer the depth. Not only is this depth map more readily available than in monocular systems, but the known baseline also allows the depth to be known to a scale. In monocular systems, an object that is twice the size of another but also twice as far away will look the same. This means that unless an object of known scale is used to calibrate the system then the system will need to run at some arbitrary scale factor. A further issue for monocular systems from this is that scale drift can be difficult to detect. This issue was partially solved in LSD-SLAM by estimating similarity transforms between keyframes to account for the drift [92], but the issue was completely eliminated by adding a stereo camera pair to their system to provide known scale [107]. This is a common trend in VO and SLAM seen in some of the most popular systems such as LSD-SLAM [92], DSO [93], and ORB-SLAM [100] where their initial formulations are designed for monocular systems and must manage and account for the unknown scale and drift, then later work expands the systems to include stereo cameras as an input [107–109].

A related input type to the stereo camera pair is the RGB-D camera. These cameras use structured IR light which is projected into the scene and the reflected back to a sensor which allows per-pixel depth of a known scale to be directly estimated [110]. They can be used in a similar way to stereo cameras, but also allow for a unique approach to VO. The approach similar to stereo camera pairs can be seen in ORB-SLAM2 [109] where the authors develop their model to treat the two input types as the same. The way that they do this is by preprocessing the input and using the RGB-D depth map to produce a virtual stereo coordinate that would have been found through point matching between a stereo camera pair. This allows the remainder of their system to be agnostic to the specific input type. RGB-D cameras open up more opportunities than simply an alternative to stereo camera pairs, however. Their dense depth maps allow for the generation of per-pixel point clouds which forms the basis of the tracking methods seen in KinectFusion [90], SLAM++ [95], and ElasticFusion [96]. In these systems, the depth map is used to generate a vertex and normal in the camera frame for each pixel. These point clouds are fused into the global map and concurrently the pose of the camera is determined by aligning the local and global clouds using a Gauss-Newton ICP alignment. A weakness of both these usages of RGB-D cameras is that the structured-light methods of depth map estimation can be made ineffective by bright light sources such as in outdoor usage and is also limited, for example, to a 1-4m range [110].

A further camera type found in the algorithms discussed in this section is that of omnidirectional and fisheye cameras which have fields of view (FoV) above 180° and up to 360° One of the limitations of standard cameras, which is particularly seen in feature-based methods, is the loss of accurate tracking when the view is dominated by texture-less regions. Direct methods remedy this by utilising all parts of the image so that the texture-less regions can still contribute some amount to tracking accuracy. Omnidirectional cameras instead aim to solve this issue by having more of the scene in view at a given time, hoping that this means suitably-textured regions remain visible throughout the tracking. Whilst monocular, stereo, and RGB-D camera systems generally use the pinhole camera model for projection, omnidirectional cameras need an alternative model. For omnidirectional SVO [111], the authors adopt the model of Scaramuzza et al. [112] which models the projection function as a polynomial. Omnidirection LSD-SLAM [113] and DSO [114] instead use the unified omnidirectional camera model proposed in the former which projects Euclidean camera coordinates first onto a unit sphere and then into the image plane. The advantage of their model is that the inverse projection function can be expressed in a closed-form.

In addition to different camera types, systems can also be built to utilise IMU data in what is referred to as visual-*inertial* odometry (VIO). Much like how a stereo or RGB-D camera system provides more direct information about scene depth, the inclusion of IMU data provides more direct information on the actual odometry itself. This odometry, however, is uncertain to a degree and is therefore fused with the visual-based estimations. One such approach to VIO is seen in MSCKF [91] which keeps a state vector comprised of a sliding window of poses from the IMU. A new snapshot of the IMU augmented to the state vector for every new frame and features are tracked in each frame to find point tracks. These tracks are used in the update step of the filter once the feature point has gone out of view. An alternative approach to VIO is referred to as *tightly*-coupled and is where error terms from the visual system and the IMU are used together in a joint optimisation problem, utilising all correlations between different terms. This is more computationally expensive, but is increasingly being used in modern systems. VIO systems such as OKVIS [115], ROVIO [105], VI-DSO [116], ORB-SLAM3 [117], BASALT [103], and Kimera [118] all use tightly-coupled VIO formulations. Despite all of these algorithms having vastly different visual odometry bases, they all ultimately use optimisation for tracking in which they can directly insert the IMU error for a joint optimisation problem.

A summary of this wide range of input types is seen in Table 2.1, showing that many systems cover a multitude of these types and paradigms. A singular entry is given in the table for systems that have initially been presented using monocular cameras and later expanded to support stereo, RGB-D, or omnidirectional cameras. A new entry is given, however, if the later addition was to covert it to a VIO system. As mentioned, it is a common trend for VO and SLAM systems to initially be designed solely for monocular camera setups and then be later expanded for further input types. This is due to the prolific availability of high quality commodity monocular cameras. However, as these systems become mature it is beneficial to solve the issues of monocular cameras, namely unknown scale and reliance on temporo-stereo depth, by introducing these more sophisticated input types.

Table 2.1: Visual odometry and SLAM systems, with their main paradigms. Classification as VO or SLAM is based on how the authors refer to their own systems. Systems whose original work was extended to accept additional input types without any other major changes have been included as singular entries, except for changes from visual to visual-inertial systems.

				Input Types						
Method	Type	Paradigm	Density	Mono	Stereo	RGB-D	Omni	IMU	Semantics	Source
MonoSLAM [88, 119]	SLAM	Indirect	Sparse	✓	-	-	-	-	-	[120, 121]*
PTAM [20, 122, 123]	SLAM	Indirect	Sparse	✓	-	-	-	-	-	[124]
DTAM [89]	SLAM	Direct	Dense	✓	-	-	-	-	-	[125]*
KinectFusion [90]	SLAM	Indirect	Dense	-	-	✓	-	-	-	[126]*
MSCKF [91, 127]	VO	Indirect	Sparse	✓	✓	-	-	✓	-	[128, 129]*
SLAM++ [95]	SLAM	Indirect	Dense	-	-	✓	-	-	✓	-
LSD-SLAM [92, 107, 113, 130]	SLAM	Direct	Dense	✓	✓	-	✓	-	-	[131]
SVO [102, 111]	VO	Hybrid	Sparse	✓	✓	-	✓	-	-	[132]
ORB-SLAM [100]	SLAM	Indirect	Sparse	✓	-	-	-	-	-	[133]
ElasticFusion [96]	SLAM	Hybrid	Dense	-	-	✓	-	-	-	[134]
OKVIS [115]	VO	Indirect	Sparse	✓	✓	-	-	✓	-	[135]
ROVIO [105, 136]	VO	Direct	Sparse	✓	-	-	-	✓	-	[137]
ORB-SLAM2 [109]	SLAM	Indirect	Sparse	✓	✓	✓	-	-	-	[138]
DSO [93, 108, 114]	VO	Direct	Sparse	✓	✓	-	✓	-	-	[139]
VI-DSO [116]	VO	Direct	Sparse	✓	✓	-	-	✓	-	[140]*
DSM [141]	SLAM	Direct	Sparse	✓	-	-	-	-	-	[142]
ORB-SLAM3 [117, 143]	SLAM	Indirect	Sparse	✓	✓	✓	✓	✓	-	[144]
BASALT [103]	SLAM	Indirect	Sparse	-	✓	-	-	✓	-	[145]
Kimera [118]	SLAM	Indirect	Sparse	✓	✓	-	-	✓	✓	[146]

* Open source implementation provided by different author.

2.4.2 Generalised keyframe VO and SLAM architecture

Visual odometry and SLAM systems can be divided into two types of architectures – filter-based and keyframe-based. A filter-based design processes each incoming frame and incorporates this into an incremental estimation of the state, which combines both the localisation and mapping, using various filters such as EFKs or particle filters. Keyframe-based designs instead periodically assign a new frame as a 'keyframe' which is then used as a reference for subsequent frames. Non-keyframes are tracked relative to the most recent keyframe or a set of active keyframes and then have their mapping data incorporated into the keyframes. This process reduces the computational complexity of any local or global optimisation processes which are performed only on the keyframes and are done to refine both the motion and the structure. Keyframe SLAM is also known by a few alternative names – fixed-lag smoothing, generally named so in comparison to filter SLAM which updates its state every frame; optimisation-based SLAM, due to the keyframes being used to make nonlinear optimisation tractable; or graph SLAM, due to the fact that the keyframes are most often used as nodes in a pose-graph representation of the map.

A modern keyframe-based approach operates with three threads; a tracking thread which has real-time requirements to initialise and determine the pose of a new frame before the next arrives, a local mapping thread which controls keyframe logic and initialises map points based on the tracking result, and a global mapping thread for loop closure and refinement which manages the long-term consistency of the map. The first two threads are common in most VO systems, whereas the third thread contains the features that elevate a system to that of SLAM. There is also sometimes a fourth thread if more expensive forms of global optimisation are included as this can take a substantial amount of time and would otherwise block the progress of other aspects.

PTAM [20] was the first to divide the tracking and mapping stages into two separate threads. The reasoning for this was that tracking needs to be done on a per-frame real-time basis, however, mapping was a slower and more computationally expensive process that did not need to be done in real-time. The introduction of the third thread has been seen in more robust SLAM systems which perform loop closure, relocalisation, and expensive global optimisation. These steps can be even more computationally expensive and are done on data over a larger time-frame and, therefore, can be separated from the tracking and local mapping.

A graphical overview of this general keyframe-based VO or SLAM architecture is given in Figure 2.14, with the major steps in each thread given. The tracking thread takes each incoming frame, extracts any intermediate data representation that might be used in the case of feature-based methods, associates this data with the current keyframe, calculates and refines a local pose estimations, and finally decides whether the new frame should become a new keyframe. In the local mapping thread any frames marked as new keyframes are added to the active set and frames not destined to be keyframes are instead optionally used to refine this set. The local mapping thread then manages the creation and removal of landmark points from the active set and performs a local optimisation before determining if any keyframes should be deactivated, dropped, or marginalised. Finally, the global mapping thread is in charge of managing the pose-graph representation of the map, detecting loop closures, optimising the pose-graph, and performing any other global optimisation. Some systems separate global bundle adjustment due to its computationally expensive nature. The global map can also be utilised for relocalisation if tracking is lost, however, this might be a part of the tracking thread or a part of the global mapping thread but block the tracking thread. As mentioned in Section 2.4.1, the distinction between VO and SLAM is usually decided based on the inclusion of long-term mapping features from the right half Figure 2.14.

The pipeline of a filter-based approach can be usefully compared to the pipeline of a keyframe-based approach, however, there are some differences that should be noted to make the comparison meaningful. Firstly, in most filter-based methods the current pose and active landmarks are managed in the state vector which usually leads to a combined tracking and mapping stage. Although some aspects of this combined tracking and mapping might be separated out, it does mean that filter-based approaches don't use the same threaded structure. Secondly, most filter-based methods only track the current pose of the robot which is incremented with each new frame and, therefore, lack a concept of a keyframe all together. This is not universally the case, however, and some filter-based methods track a window of poses with which they use very similar logic to keyframe-based methods as to when poses enter and leave this window.

The remainder of this section compares the features of the systems given Table 2.1 based on major sections of the architecture given in Figure 2.14. This comparison is summarised in tabular form in Tables 2.2-2.4.

Local tracking

The most basic function of visual odometry is to determine the pose of the latest image frame with respect to a prior frame. In keyframe VO and SLAM, this is with respect to either the latest keyframe or a window of active keyframes. It is in this stage of the pipeline that the paradigm distinction between *direct* and *indirect* has the most significance.

The traditional indirect method involves finding point correspondences which are then used to determine the pose. The way in which the correspondences are de-

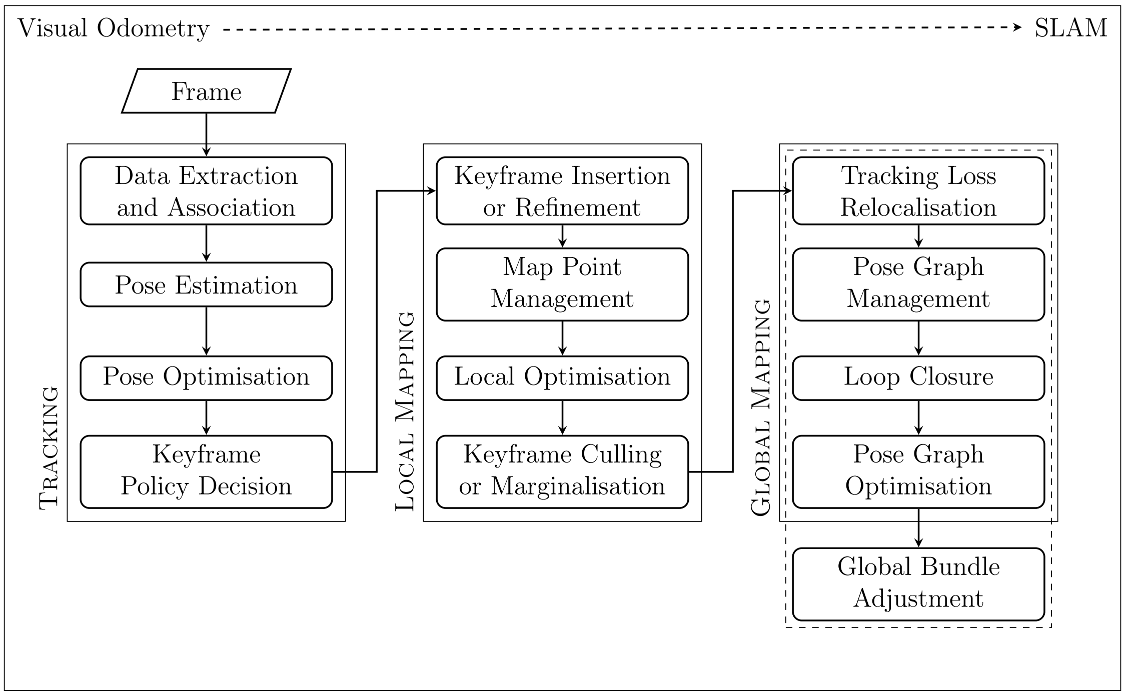


Figure 2.14: A generic keyframe SLAM framework, divided into three (or optionally four) threads. The tracking thread operates at frame-rate, the local mapping thread operates on keyframes, and the global mapping thread has the lowest priority in terms of real-time operation. If global BA is included, it can be separated into its own thread due to the processing time required.

termined is dependent on how the points to be matched are represented. A common feature point representation is simple image patches. Methods such as MonoSLAM [88] and MSCKF [91] use gated searches for image patch descriptors compared using normalised cross-correlations, whilst PTAM [20] uses similar image patches that are instead compared with zero-mean sum of squared differences (SSD). More recent methods that use image patches as feature descriptors tend to instead determine correspondences using the Kanade-Lucas-Tomasi feature tracker approach which aligns the patches iteratively using gradient descent on the photometric error between windows [98, 147, 148]. Methods that use this approach include SVO [102], BASALT [103], and Kimera [118]. Many indirect methods instead use robust feature descriptors such as SIFT [149], SURF [150], BRISK [151], BRIEF [152], or ORB [101] descriptors. These descriptors are designed to efficiently describe the local region surrounding a keypoint in a manner that remains consistent after scale, rotation, or small lighting changes. Such features are compared using Euclidean distance for vector-based features and Hamming distance for binary features, and correspondences can efficiently be determined using approximate nearest-neighbour approaches [153]. Once these different approaches have been used to obtain a set of correspondences, most systems then proceed to determine the pose using the Perspective-n-Point (PnP) algorithm with Random Sample Consensus (RANSAC)

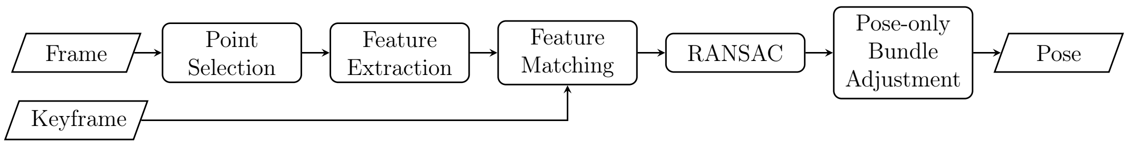


Figure 2.15: The local tracking process in feature-based indirect systems, determining the pose between a new frame and an earlier frame or keyframe.

[104]. This process assumes that some percentage of the correspondences are falsely matched outliers and repeatedly solves the PnP problem with a random selection of points then tests how many of the full set of points agree with the estimate. The process continues until either a sufficient percentage agree or else it takes the best estimate from a fixed number of tests. This estimate is then usually refined through a nonlinear least squares problems that optimises the pose by minimising the reprojection error between the matched features and the projections of their triangulated 3D points, which is essentially a single-pose bundle adjustment [154]. This process is summarised in Figure 2.15.

An alternative approach to tracking taken by dense RGB-D systems involves the alignment of point clouds in order to determine new poses. In an RGB-D system, each new frame provides both a colour image and a depth image and, based on the calibration of the camera, this depth image can be projected out to create a per-pixel point cloud. The pose of the camera can be determined by aligning the local point cloud with a reconstructed view of the global map or a previous local map using a gradient descent Iterative Closest Point (ICP) algorithm. KinectFusion [90] does this by generating a vertex map from the depth map and then filtering over this to generate a normal map. These maps are built into a pyramid using different pyramid levels of the depth map which allows for it to be aligned with the corresponding maps from the previous frame in a coarse-to-fine approach. SLAM++ [95] follows a similar method, however, they semantically label their global map by fitting meshes with known objects and it is this global map that is projected into the last frame against which the vertex and normal map is aligned. They argue that provides a higher quality representation of the scene at early stages in tracking when KinectFusion's map is still incomplete. ElasticFusion [96] is another system that follows a similar approach to the previous two, however, they use what they refer to as a 'surfel' which combines a vertex, normal, colour, weight, radius, and timestamp. Therefore, their ICP process is done as a joint geometric and photometric alignment. This local tracking procedure is summarised in Figure 2.16.

Direct methods take a single-step approach of directly aligning the current frame with a previous one. This is done using the iterative Lucas-Kanade (LK) algorithm [147], which is usually done with the more computationally efficient formulations of either Forward Compositional LK (FC-LK) or Inverse Compositional LK (IC-

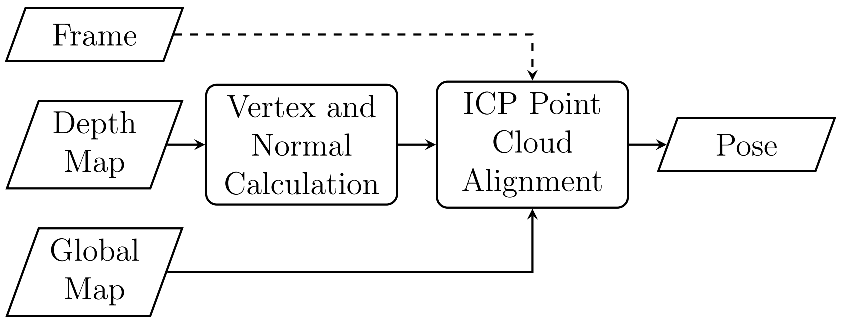


Figure 2.16: The local tracking process in RGB-D systems, determining the pose between the point cloud of a new frame and that of the global map. Some methods, such as ElasticFusion [96], utilise photometric data from the image frame in the ICP alignment.

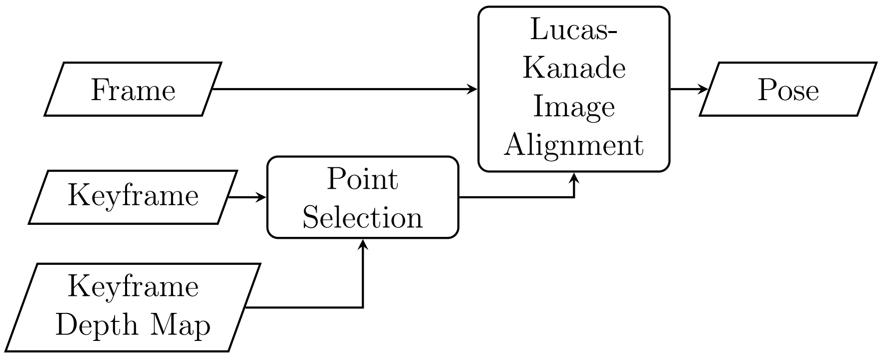


Figure 2.17: The local tracking process in direct systems, determining the pose between a new frame and an earlier frame or keyframe.

LK) methods given by Baker and Matthews [106]. These methods involve warping a template image, usually the reference keyframe, into the current frame according to the current estimate of a rigid-body transformation as well as an already estimated depth map. The method uses gradient descent to optimise the transformation estimation by minimising the photometric error of the raw intensity. DTAM [89] is one such algorithm that performs a dense coarse-to-fine FC-LK alignment to determine poses. LSD-SLAM [92] also performs a semi-dense coarse-to-fine FC-LK alignment, however, they use variance-normalised photometric error propagated from the depth map estimation. They also use rigid-body transformation in frame-to-keyframe tracking and a similarity transform for keyframe-to-keyframe tracking. DSO [93] departs from the previous two approaches in two main ways; firstly, they moved to a sparse formulation that selects 2000 evenly spread but high gradient points, and secondly, they applied an affine lighting model to reduce the effect of brightness inconsistency between corresponding pixels. SVO [102] is a hybrid approach that begins with a sparse version of the alignment done in DTAM, but then uses a KLT feature tracker to gain subpixel correspondences that are then used in the standard indirect method. The general direct tracking method is summarised in Figure 2.17.

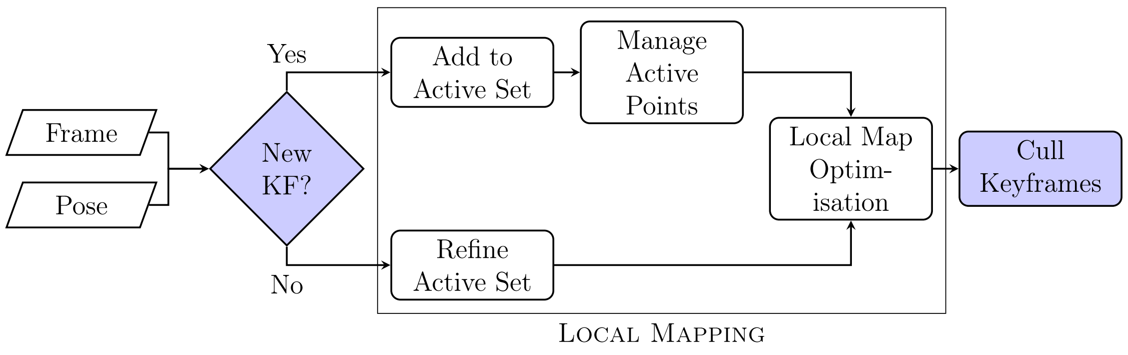


Figure 2.18: The local mapping thread from the point of view of keyframe creation and culling, which are highlighted in blue. The rest of the local mapping process includes either adding to or refining the active set of keyframes, managing active points, and local optimisation.

Keyframe policies

The defining feature of keyframe SLAM is that it performs optimisation on keyframes. That is, only a subset of frames are elevated to keyframe status which form both the window of poses and associate structures included in local optimisation as well as nodes in the graphical structure used in pose graph optimisation and global optimisation. Each VO and SLAM system has its own logic relating to when a keyframe is created from a frame, and also has logic for when a keyframe is culled. Culling can involve simply dropping it from the system, marginalising it out, or removing it from the active window but keeping it for possible reactivation later. These processes of creating and culling keyframes essentially bookend the local mapping thread, as demonstrated in Figure 2.18. The possibility of creating a keyframe is tested on a per-frame basis; if it is successful the frame is added to the active set and incorporated into the map, and if it is unsuccessful then the non-keyframe is often used to refine map point estimates related to the active set. The test for culling of any keyframes is often done after local map optimisation, but might be performed as a combined step with keyframe creation as soon as the active set has exceeded its target size.

Not all of the systems considered here use keyframes; their analysis in the given keyframe SLAM framework is simply a means to compare them to those systems that do. The EKF-based SLAM systems including MonoSLAM [88], MSCKF [91], and ROVIO [105] all incorporate every new frame into the filter state. However, the way that MSCKF tracks a window of poses in its state is similar to the keyframe logic of other systems, particularly in that it doesn't simply remove the oldest state when the window is full but instead removes a third of the poses equally spaced in time. This could be considered as a keyframe policy where all frames are keyframes and culling is done to maintain a good temporal spread. Similarly, the dense RGB-D SLAM methods of KinectFusion [90], SLAM++ [95], and ElasticFusion [96] all fuse every image frame into the global map and lack any feature like a keyframe.

Keyframe creation is often done based on factors such as number of frames since last keyframe, time elapsed, distance travelled, rotation undertaken, covisibility of pixels or active points with previous keyframe, or visibility of map points in the current frame. PTAM [20] adds a new keyframe once at least twenty frames have elapsed since the last keyframe creation, but also only if the tracking on the new frame is good and the camera has moved a minimum distance. ORB-SLAM [100] similarly inserts a keyframe when at least twenty frames have elapsed, but they also test for whether the current frame tracks less than 90% of the points in the reference keyframe. The authors also state that they insert new keyframes as fast as possible because they also have a mechanism to cull redundant keyframes. SVO [102] has a similar visibility requirement to ORB-SLAM, however they base theirs on the visibility of 3D map points rather than points in the previous keyframe. OKVIS [115] uses a combination of these two approaches; they insert a keyframe if the hull of projected landmarks fills less than 50% of the image or if fewer than 20% of keypoints are matched in the latest frame. Like with PTAM's keyframe creation based on distance, other systems also follow a similar approach. LSD-SLAM [92] has separate limits for distance travelled and rotation undertaken, which can be thought of as serving the same purpose as the visibility-based decisions of the previously discussed systems. DSO [93] does the same translation-based decision, but replaces the rotation with a direct comparison of field-of-view overlap. They also create new keyframes when there has been a significant change in exposure time. A final useful measure in keyframe creation is the real time elapsed since the last keyframe. This differs from number of frames elapsed as it is more of a consideration of VIO systems. In VI-DSO [116], they point out that the IMU pre-integration between keyframes becomes more inaccurate as time passes, so they make sure that a keyframe is created at least every 0.5 seconds. This is also done in ORB-SLAM3's visual-inertial version [143] where they ensure keyframes are never more than 0.5 seconds apart.

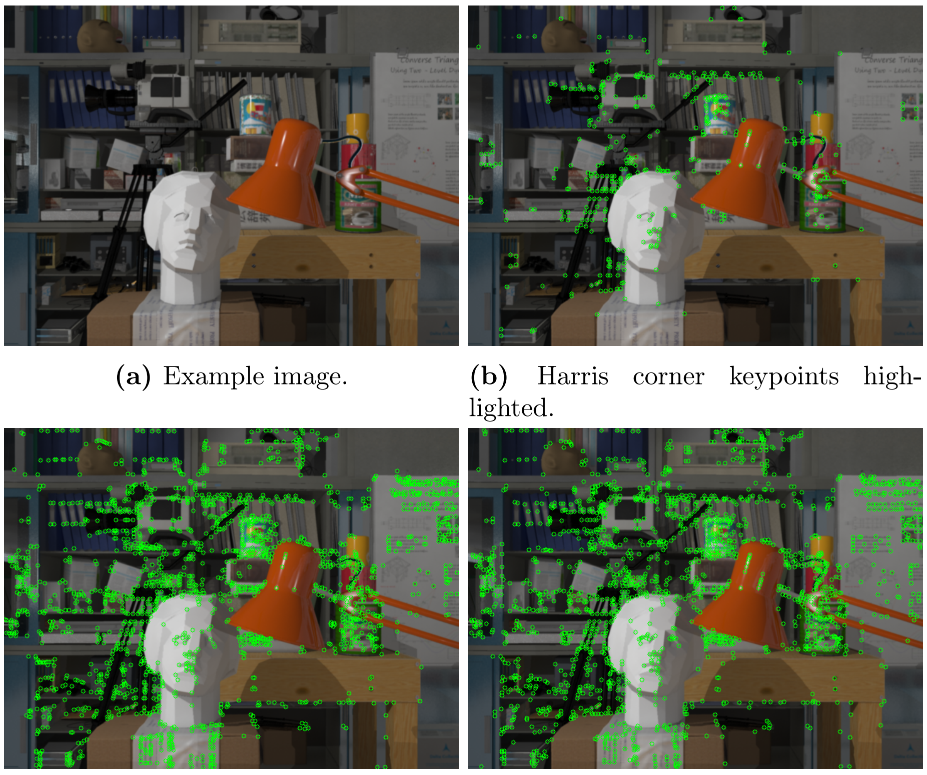
The other aspect to keyframe management is when to remove a keyframe from either the active window or the system entirely. This can be done by dropping the data, deactivating it for later, or marginalising it out. MSCKF, for example, upon reaching its frame limit drops one third of the frames once all of their feature observations have been utilised and an EKF update has been done. DSO, in comparison, performs marginalisation on the keyframe before removing it from the active window. Firstly the points in the frame are marginalised, then the frame itself is marginalised. They drop any terms relating to points that would adversely affect the sparsity of the Hessian. When choosing which keyframe to marginalise, they don't simple remove the oldest keyframe. Instead they test for covisibility and aim to have good coverage across 3D space. DSM [141] is a SLAM system that uses similar methods to DSO, however, as it uses pose graph optimisation and loop closure it does not want to marginalise out keyframes as they leave the active window. What it does instead is temporarily drop older keyframes, but then reactivates them for loop closures. ORB-SLAM also does not marginalise out old keyframes due to their usage in the pose graph, however, it does detect and discard redundant keyframes that have 90% of their points visible in at least three other keyframes.

Point policies

Each sparse system has its own method for determining which points in an image get used. Dense systems do not consider this because they always use the full image, however, in sparse systems the goal is to use the points that will lead to the best tracking.

Many of the systems discussed in this section select their points using the Harris corner detector [97], Shi-Tomasi corner detector [98], or the FAST feature detector [99]. The Harris and Shi-Tomasi corner detectors score pixels based on the eigenvalues of their second-moment matrices, allowing the selection of points where the image has the most change in all directions. These points are considered suitable for feature extraction as they are the most distinctive between images regardless of scale and rotation changes. The FAST feature detector aims to find these corners through a simplified approach that instead uses a small number of pixel comparisons. These three corner detectors are among the most common used in SLAM, however some systems that rely on ORB features instead use the ORB feature detector [101]. The ORB detector uses a modified FAST feature detector that finds the best points from FAST by computing the Harris score, then also computes an orientation by finding the intensity-weighted centroid of the local area. A comparison of Harris corners, Shi-Tomasi corners, and FAST features is given in Figure 2.19.

Some direct methods have a very different approach to selecting which points to use. A weakness of corner detectors is that they return very few candidate points in images of flat or low-texture regions. Sparse direct methods have their roots in dense direct methods which seek to utilise all of the image to avoid the issues associated with low-texture regions, so these methods aim for a good spread of points across the image regardless of texture level. However, they do still aim to have most of their points in the high-texture regions. This can be seen in DSO [93] and related methods. Their approach is to divide the image up into a grid based on how many points are sought and then select the point in each block with the highest gradient over a threshold. They do this iteratively while lowering the threshold to get some points that are in the low-texture regions.


(a) Example image.

(b) Harris corner keypoints highlighted.

(c) Shi-Tomasi corner keypoints highlighted.

(d) FAST feature keypoints highlighed.

Figure 2.19: Comparison of feature keypoint detection for Harris corners, Shi-Tomasi corners, and FAST features.

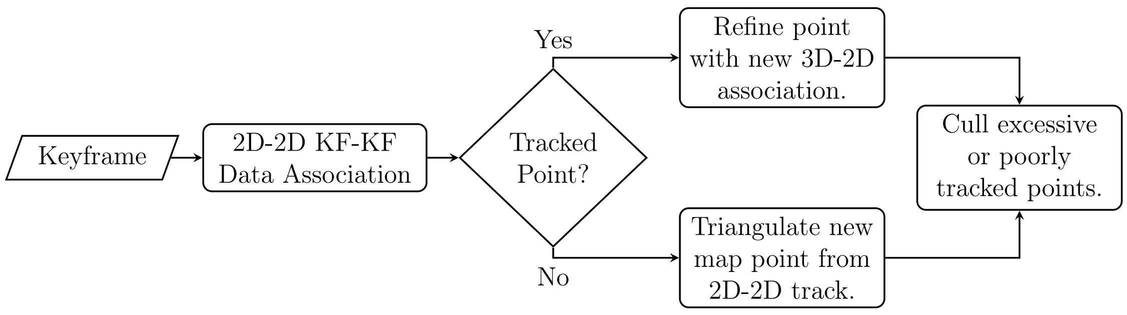

Keyframe
2D-2D KF-KF Data Association
Tracked Point?
Yes
No
Refine point with new 3D-2D association.
Triangulate new map point from 2D-2D track.
Cull excessive or poorly tracked points.

Figure 2.20: Local mapping in indirect systems. Feature points in a new keyframe are associated with older keyframes. If they relate to an existing 3D map point the association refines the estimate, if it is a new map point then that is triangulated from the point track.

Local mapping

The mapping stage is where the point data from a frame or keyframe is incorporated into the global map once the relevant pose has been determined by the tracking stage. Most keyframe VO and SLAM systems alternate between tracking and mapping, however, it is not necessarily a one-to-one alternation. Instead, the mapping is done in a separate thread without real-time requirements. The local mapping itself is one of a handful of responsibilities of the mapping thread, which also includes keyframe and map point management as well as local optimisation.

Indirect feature-based methods generally use the correspondences and pose to determine 3D point locations. SVO [102], ORB-SLAM [100], OKVIS [115], BASALT [103], and Kimera [118] triangulate the 2D-2D correspondences and add the resulting 3D points to the global map. SVO does this once the associated variance is sufficiently low, whereas ORB-SLAM first does a local bundle adjustment for all points in the current keyframe. This process is summarised in Figure 2.20, showing new 2D-2D correspondences leading to new points and correspondences with points already on the map being used for refinement.

Direct methods don't have explicit correspondences from their pose tracking and therefore need to perform a guided search. These methods generally parameterise the map by inverse depth. That is, rather than storing the map as a set of 3D points in the global reference frame, they instead store it as the inverse depth at a given pixel corresponding to the 3D point. Since the 2D coordinates of these reference points are fixed, this means that during bundle adjustment only one parameter per map point needs to be refined. This inverse depth is often found by performing epipolar searches that provide constrained correspondences. Although PTAM [20] is an indirect method and initialises the map with a RANSAC approach like other similar methods, when it is searching for already mapped points the map is projected into the current frame and a gated epipolar search is performed around the expected

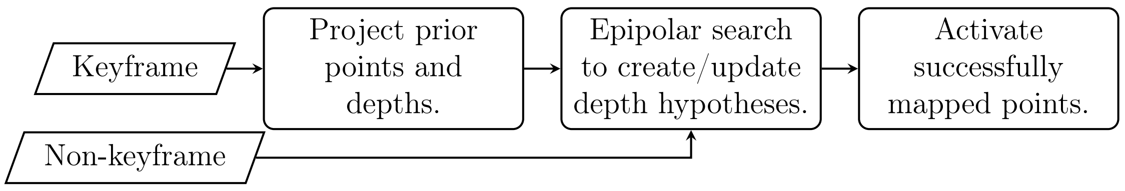

Keyframe
Non-keyframe
Project prior points and depths.
Epipolar search to create/update depth hypotheses.
Activate successfully mapped points.

Figure 2.21: Local mapping in direct systems, such as DSO [93]. A map point is represented by a depth hypothesis associated with an image point. Depth hypotheses are created through 1D epipolar searches, which are then used to narrow the search range when later refining those searches. Keyframes are initialised by projected points from older keyframes.

location. LSD-SLAM [92] attempts to perform epipolar searches for every pixel in the keyframe against new frames, however, it aborts early if the gradient isn't of a sufficient quality. It also limits the search range to twice the variance if the given keyframe point already has a depth hypothesis. DSO [93] performs a similar search, but it then refines the match with an iteration of gradient descent which is similar to the subpixel correspondence relaxation seen in the KLT tracker of SVO. DTAM [89] is different to other direct methods in that it formulates the dense depth map estimation as a primal-dual minimisation of a regularised photometric cost. A process similar to that found in LSD-SLAM and particularly DSO is summarised in Figure 2.21. New keyframes can be initialised by projecting prior depths into the new frame, then subsequent non-keyframes are used to create and update depth hypotheses from the epipolar search. DSO also activates a subset of the successfully tracked points for use in local optimisation.

The local mapping stage of dense RGB-D methods is different to other indirect and direct methods. It involves fusing the local point cloud into the global point cloud map. KinectFusion [90] does this by optimising over a volumetric truncated signed distance function to fuse the surfaces together. ElasticFusion [96] does this fusion simply by bringing the colour and depth maps of the live frame into close alignment with the global map and associating these points with the global surfels. SLAM++ [95] has a fairly different approach where they leverage their semantic labelling. Their map is comprised of known objects and meshes which are aligned with the live point cloud through an active search and then verified by repeating ICP alignment back into the live camera view.

Local map optimisation

Keyframe VO and SLAM systems often have a local window of keyframes which represent the 'active' portion of the map which undergoes local optimisation. Each incremental tracking estimation and local mapping process introduces an accumulation of error in the system that causes it to drift, which is greatly reduced by optimising over a number of keyframe poses and structure. As keyframes are added and removed from the active region this can be seen as a sliding-window optimisation and is done in the form of local bundle adjustment or pose-graph optimisation.

Local bundle adjustment is the most common form of this optimisation which is done by minimising the reprojection error of the map points. This is generally of the form given in Equation 2.12 where ξ is a pose in the set of keyframes $\mathcal{F}$, $\mathbf{X}$ is a 3D map point in the set of map points $\mathcal{X}$, $\mathbf{x}_{ij}$ is the feature location in keyframe i associated with map point j, obs$(\cdot)$ is the set of keyframes in which a given map point has been observed, and $\pi(\cdot)$ projects a 3D map point into the keyframe. As can be seen, the reprojection error is the error between where the point is projected into the image and the associated feature location, which is usually less than one pixel. Although many full SLAM systems avoid performing global bundle adjustment due to the prohibitive processing time it can often take, many systems including VO systems perform local optimisation. By only optimising over the motion and structure of a small window of keyframes the accuracy of the estimation can be greatly improved without the excessive processing time. PTAM [20], through introducing the mapping thread and keyframe concepts, enabled the idea of local BA on the active window. In their approach, they have two sets of keyframes; the most recent keyframes and the next set of most recent keyframes before them, numbered at three and seven respectively in their example. The latter set has their parameters fixed and the former set has its motion and structure refined using estimations from both sets. The approach seen in SVO [102] performs three steps of BA. They first perform pose-only BA on the newest keyframe, then structure-only BA on its map points, and finally a full local BA on the motion and structure of the newest keyframe and all nearby keyframes. ORB-SLAM [100] also does local BA on an active window, however, they also include any keyframes that are co-visible with the active window.

$$\{\xi_i^*, \mathbf{X}_j^* \mid i \in \mathcal{F}, j \in \mathcal{X}\} = \underset{\{\xi_i, \mathbf{X}_j\}}{\arg\min} \sum_{j \in \mathcal{X}} \sum_{i \in \text{obs}(j)} \frac{1}{2} ||\mathbf{x}_{ij} - \pi(\mathbf{X}_j; \xi_i)||_2^2 \tag{2.12}$$

This local BA is also done in VIO systems, however, these systems jointly minimise reprojection error and IMU error. This can be seen in OKVIS [115] and BASALT [103], which both optimise over windows of the most recent keyframes. ORB-SLAM3 [117] uses the same approach as the earlier ORB-SLAM, including co-visible keyframes to those of the active window, however in its VIO mode it also performs the joint minimisation of reprojection error and IMU error.

Closely related to the standard form of bundle adjustment seen in feature-based methods is the *photometric* bundle adjustment found in direct methods. While standard *geometric* BA is minimising reprojection error, photometric BA is minimising the photometric error. It is essentially an expanded form of the direct tracking problem, here instead being optimised for both pose and structure for a window of keyframes. Conversely, the direct tracking problem can be seen as pose-only photometric BA being performed between only two frames. This optimisation problem is given in Equation 2.13. Using mostly the same notation as Equation 2.12, the difference here is that instead of optimising 3D map points this optimises a depth map, D, associated with a keyframe. Additionally, I is an image, Ω is the set of tracked points in a keyframe, and $w(\cdot)$ warps a point from one image to another according to the pose of the keyframe and depth of the point. Therefore, this problem is using the error between corresponding pixel intensities across all frames that the tracked pixel has been observed. DSO [93] performs this sliding-window photometric BA on its active window which is simply the set of most recent keyframes. DSM [141] uses a similar approach, however they also include older reactivated keyframes from their loop closure.

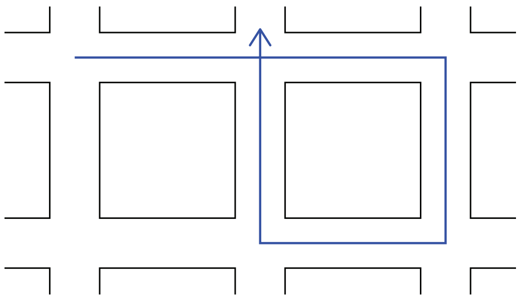
$$\{\xi_i^*, D_i^* \mid i \in \mathcal{F}\} = \underset{\{\xi_i, D_i\}}{\arg\min} \sum_{i \in \mathcal{F}} \sum_{\mathbf{x} \in \Omega_i} \sum_{j \in \text{obs}(\mathbf{x})} \frac{1}{2} ||I_i(\mathbf{x}) - I_j(w(\mathbf{x}; \xi_i, D_i(\mathbf{x})))||_2^2 \quad (2.13)$$

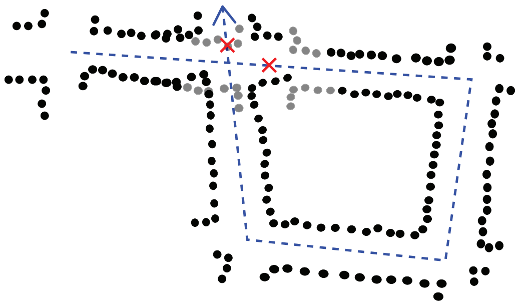
There are alternative methods used to perform local optimisation. ElasticFusion [96] performs local map maintenance by searching for short-term loop closures through comparisons of the local portion of its map to older portions. Alternatively, Kimera [118] performs factor graph optimisation on a local graph that is separate from its global map.

Loop closure and relocalisation

Two of the most common features that lead to long-term map accuracy and elevate VO systems to be considered SLAM are loop closure and relocalisation. These two features are closely related and often implemented in the same way – although not always. Loop closure refers to the recognition that two places separated in time are spatially the same place, which allows for constraints to be added to the global map that eliminates the drift that might have occurred in the time between those two poses. This process is illustrated in Figure 2.22. Relocalisation is essentially the same process but done in a different context; when tracking is lost for any reason a SLAM system will attempt to find the best keyframe with which to reestablish tracking.

One of the most common forms of loop closure is done with appearance-based methods such as bag-of-visual-words. These methods require a ‘code-book’ that is generated offline based on clusters of the desired feature type learnt from an


(a) Example scene with ground truth path.


(b) Estimated path and map before loop closure.

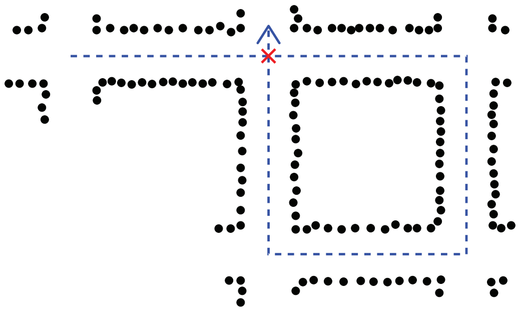

(c) Estimated path and map after loop closure.

Figure 2.22: An example scene showing loop closure. The estimation of the path and point cloud of the map have inaccuracies and drift over time. The ground truth is show in (a). In (b), the red crosses show two points that are the same location, and the grey map points are points that have been re-detected at different locations. After loop closure in (c) the two points are identified as the same location, allowing the path and map to be updated.

image set similar to the domain in which the SLAM system will operate. This allows the online SLAM system to efficiently represent images as bags of these code-words which are based on code-word occurrence but not their location in the image. Images can therefore be compared probabilistically across appearance-space and be robustly matched even with significant changes to lighting [155]. Popular implementations of bag-of-visual-words include FAB-MAP 2.0 [21], openFABMAP [22], and DBoW2 [23]. LSD-SLAM [92] utilises openFABMAP for its loop detection and adds these constraints to its global pose graph. This reveals one limitation seen in direct methods where feature-based methods are still required for long-term map accuracy. ORB-SLAM [100] and Kimera [118] both use DBoW2 for their loop closure which is built around the ORB descriptors that they both use for other parts of their SLAM pipelines. For a given keyframe ORB-SLAM uses DBoW2 to find other keyframes that are more similar to it than its least similar co-visible keyframe. The potential matches are then accepted as loop closures when three co-visible keyframes are proposed. Kimera verifies its potential matches through geometric means in its front-end tracking.

There are many alternative methods for loop closure. SLAM++ [95] detects loops by matching fragments of its global point cloud map with other regions of its full map using predictive ICP. DSM [116] has a basic form of loop closure where it reactivates earlier keyframes into its active window if it predicts that they will fill in gaps in the field of view of the current keyframe. Their method, however, assumes that the drift in the loop is not significant enough to push the keyframe out of the predicted field of view. BASALT [103] also has a simple form of loop closure where ORB features are extracted and matched between keyframes which allows the addition of edges to the pose graph which are statistically independent to the KLT feature tracking done in its front end. These systems all have loop closure features that are more similar to the standard tracking methods than the appearance-based loop closure commonly seen.

Relocalisation is often implemented in much the same manner as loop closure. When tracking is lost in a SLAM system the relocalisation module is initiated to find a suitable keyframe with which to reestablish tracking. In SLAM++, when tracking is lost a new local map is generated. Once this map becomes large enough it is tracked against the global map in the same way that fragments of the global map were used for loop closure. ORB-SLAM also has similar loop closure and relocalisation methods with their DBoW2 database allowing for the current frame to be matched against candidate keyframe in appearance-space.

Not all relocalisation methods are based on the loop closure methods in a given system. LSD-SLAM [92] attempts to reestablish tracking against well-connected keyframes at random and then tests the neighbours of a successful candidate to see if they are a better match. DSO [93] does not have a loop closure mechanism, nor a global relocalisation mechanism, but it does implement a simple form of local relocalisation. If tracking is lost in DSO, then a large number of motion models with differing amounts of translation and rotation are tested as the initial hypothesis for its standard tracking procedure on the coarsest pyramid level and the model that produces the least error is taken as the correct pose.

Pose-graph and global optimisation

An important factor in making the SLAM map accurate in the long term is to utilise all estimates, including any relocalisation or loop closure terms, to perform some kind of global optimisation. This optimisation can come in two forms – bundle adjustment or pose-graph optimisation. One could perform the same kind of bundle adjustment done on the active window except globally for all poses and structure, however, this is extremely time consuming and often intractable. A much more feasible method is to perform pose-graph optimisation which refines only the poses based on the relative estimates at each keyframe.

For global bundle adjustment the cost term is comprised of error in re-projection of the 3D landmarks into each keyframe for the geometric case, as given in Equation 2.12, or photometric error between the intensity of a landmark in its reference frame compared to any other frames in which it has been located, as given in Equation 2.13. Bundle adjustment then seeks to refine the set of all poses and landmarks by minimising this error. The main difference for these two equations to their use in local optimisation is that for global optimisation the set of keyframes $\mathcal{F}$ includes all keyframes rather than just the active window. This process is extremely computationally intensive and is usually avoided. It can be made somewhat more tractable by performing pose-only BA where only the pose is refined using the same error terms. Pose-graph optimisation is closely related to BA, however, the error term is constructed differently. Here, the problem is represented by a factor graph and the error comes from the disagreement between the current global pose estimates and relative pose measurements between keyframes. A representation of this factor graph representation of the full SLAM problem is given in Figure 2.23. The poses at each keyframe, landmark positions, and intrinsic calibration are represented as nodes, and the relative measurements between nodes are factors represented by small squares. Like with pose-only BA, pose-graph optimisation usually only includes the nodes related to keyframe poses and the factors between them. This optimisation problem is expressed in Equation 2.14, again using the same notation as in Equations 2.12-2.13, with $\mathcal{E}$ being the set of all edges in the graph and $\Delta\xi$ being a relative pose between two nodes. These relative poses correspond to the edge factors in Figure 2.23 – the $\mathbf{u}$ factors for poses between two immediate neighbour nodes and the $\mathbf{c}$ factors for

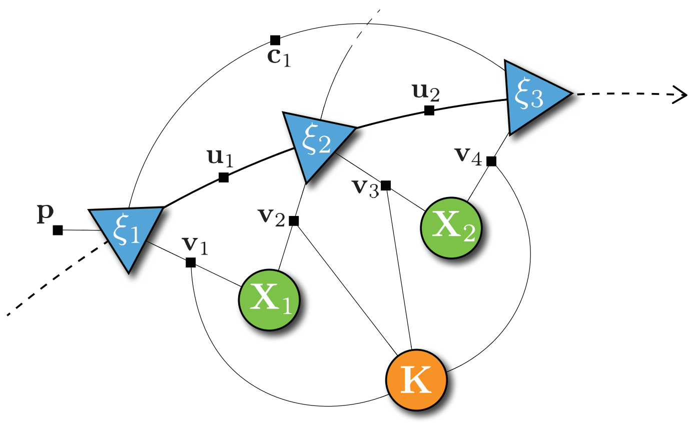


Figure 2.23: A full factor graph representation of a SLAM map. ξ_1-ξ_3 are poses of the robot at times 1 to 3, $\mathbf{X}_1$ and $\mathbf{X}_2$ are landmarks seen by the robot at various times, and $\mathbf{K}$ is the intrinsic calibration of the robot's camera. $\mathbf{p}$ represents a prior factor, $\mathbf{u}_1$ and $\mathbf{u}_2$ are factors between poses. $\mathbf{v}_1$-$\mathbf{v}_4$ are factors between the robot poses, landmarks, and camera projection. $\mathbf{c}_1$ is a loop closure factor between non-immediate poses. *Pose-graph* optimisation only optimises for the robot pose nodes and only considers the $\mathbf{u}$ and $\mathbf{C}$ factors between them.

poses between nodes in loop closures. In a system where the pose-graph is a tree, perhaps due to a lack of loop closures and keyframes only being connected with their direct predecessor, there would be no disagreement between relative measurements and the pose-graph would start at its optimum value. However, once we introduce loop closures and added edges based on co-visibility, then the system has the almost certain opportunity for disagreement. One can think of pose-graph optimisation as using the edges on the graph as springs which pull the poses in various directions until they are in the lowest energy state – the state where relative measurements agree the most.

$$\{\xi_i^* \mid i \in \mathcal{F}\} = \underset{\{\xi_i\}}{\arg\min} \sum_{(i,j)\in\mathcal{E}} \frac{1}{2}||\xi_i \Delta\xi_{ij} - \xi_j||_2^2 \tag{2.14}$$

As mentioned, global BA is a time-consuming procedure and, therefore, most systems that have a form of global optimisation rely instead in pose-graph optimisation. This can be seen in systems such as SLAM++ [95], LSD-SLAM [92], ORB-SLAM [100], and Kimera [118]. Later versions of ORB-SLAM, ORB-SLAM2 [109] and ORB-SLAM3 [117], also include global BA. To prevent this expensive procedure from blocking other features of their SLAM system, they perform this global BA entirely in its own thread. Furthermore, since loop closure can add edges to their graph which alters the optimum that the BA is working towards, they abort and restart the global BA whenever a new loop closure is identified. BASALT [103] also performs global BA based on ORB features that are extracted and matched entirely separately to the KLT matched image patch features used by their tracking and mapping subsystems.

2.4.3 Summary and open problems

This section has revealed a number of trends in the field. The two main paradigm splits in VO and SLAM are those between *direct* versus *indirect* methods, and between *dense* versus *sparse* methods, although, there exist examples of hybrid methods with features of direct and indirect methods as well as methods that fall on a spectrum between dense and sparse. The differentiation between direct and indirect methods is one of the aspects of the field that receives the most attention currently, with some of the most popular direct methods including LSD-SLAM [92] and DSO [93], and one of the most popular indirect methods being ORB-SLAM [100]. Indirect methods present a number of advantages over direct methods due to the use of easily identifiable features that allow matching across wide baselines as well as different mapping sessions. One of the disadvantages of indirect methods is that the detection of features, calculation of feature descriptors, and their matching is computationally expensive and therefore limits the hardware in which the SLAM systems can be used. This is addressed by direct methods that operate directly on the raw images or some primitive feature type close to the raw image, with matching being done through whole image alignment. Direct methods replace many of the computationally expensive components of indirect methods with more efficient alternatives and can operate in featureless scenes, however, they lose the ability to match along wide baselines and often have to resort to indirect methods for loop closure and relocalisation. Sparse direct methods such as DSO [93] are amongst the most efficient options available, whereas indirect methods such as ORB-SLAM [100] amongst the most feature-rich and robust options. Although there are examples of hybrid methods that utilise portions of direct and indirect methods, such as SVO [102], there is a need for research that aligns the two paradigms more closely. As it currently stands, the utilisation of indirect qualities within direct algorithms for wide baseline matching or loop closure requires the dedicated detection, computation, and matching of features which wastes the computational savings of direct methods if such features are not utilised elsewhere.

This section also presented a general keyframe-SLAM framework that identified the three, or optionally four, threads that comprise a full SLAM system, with VO systems focusing on the first and second threads. Relating this to the direct versus indirect paradigm split, the key components of a direct system generally focus on the tracking and local mapping thread. Considering the most feature-complete full SLAM direct method, LSD-SLAM, the qualities that make it ‘direct’ are mostly found in the tracking thread, whereas the pose graph optimisation is agnostic to the direct-indirect distinction and the loop closure relies on an indirect feature-based method, using FAB-MAP.

From this analysis, a number of open problems have been identified:

- Direct components can be incorporated into loop closure and relocalisation that could be similar to the point cloud matching methods seen in KinectFusion [90] and ElasticFusion [96].
- Since full direct SLAM methods calculate feature descriptors for loop closure, one could incorporate such features into the direct alignment pipeline to robustify that component without relying on the comparatively more expensive feature matching of indirect methods.
- The suitability of direct methods for wide-baseline matching should be explored to understand how this compares incorporating feature descriptors.

Table 2.2: Comparison of the *tracking thread* concept in Keyframe SLAM for different systems.

Method	Point Selection	Data Type	Data Association	Tracking
MonoSLAM	Shi-Tomasi corner detector.	Image intensity patches.	Normalised cross-correlation.	EKF containing latest pose and landmarks in state, with constant motion model.
PTAM	FAST feature detector with Shi-Tomasi scores, as well as edglets.	Image intensity patches.	Zero-mean SSD.	Motion model followed by pose-only local bundle adjustment.
DTAM	Dense intensity image.	Raw intensity image.	Direct image alignment.	Pyramid Lucas-Kanade image alignment, first rotation then full pose.
KinectFusion	Dense RGB-D.	Vertex and normal-based surface derived from RGB-D images.	Direct geometric alignment of surface.	ICP minimisation of the error between a predicted surface projection of the global 3D reconstruction with the observed vertex and normal map.
MSCKF	Shi-Tomasi corner detector.	Image intensity patches.	Cross-correlation.	EKF containing window of poses but no landmarks in state, with constant motion model.
SLAM++	Dense RGB-D.	Vertex and normal-based surface derived from RGB-D images.	Direct geometric alignment of surface.	ICP Gauss-Newton alignment of depth and normal map between live camera and predicted view based on current map.
LSD-SLAM	Semi-dense intensity, all points with successful depth hypotheses.	Raw intensity image.	Direct image alignment.	Keyframe-based Lucas-Kanade with variance weighted residuals.

Table 2.2: (Continued)

Method	Point Selection	Data Type	Data Association	Tracking
SVO	FAST feature detector and gradient-based.	Image intensity patches.	Kanade-Lucas-Tomasi feature tracker.	Sparse Lucas-Kanade whole image alignment followed by Kanade-Lucas-Tomasi feature tracker to relax correspondences to sub-pixel locations, finally pose-only local bundle adjustment.
ORB-SLAM	ORB feature detector.	ORB descriptor.	Hamming distance between descriptors.	Constant motion model leading to guided correspondence search, followed by bundle adjustment.
ElasticFusion	Dense RGB-D.	Vertex and normal-based surface derived from RGB-D images.	Direct photometric and geometric alignment of coloured point cloud.	Joint ICP geometric and photmetric alignment between the current frame and depth map with a predicted view.
OKVIS	Harris corner detector.	BRISK descriptors orientated with gravity.	Hamming distance between descriptors.	RANSAC on 3D-2D landmark correspondences, followed by RANSAC on 2D-2D correspondences with respect to latest keyframe.
ROVIO	Shi-Tomasi corner detector.	Pyramidal multi-level image intensity patches.	Direct image alignment.	Fully robo-centric EKF with robot position and relative landmarks as state.
ORB-SLAM2	ORB feature detector.	ORB descriptor.	Hamming distance between descriptors.	Constant motion model leading to guided correspondence search, followed by bundle adjustment.

Table 2.2: (Continued)

Method	Point Selection	Data Type	Data Association	Tracking
DSO	Candidate points are chosen to be well spaced in image and have high gradient, activated points are successfully tracked and spread in 3D space.	Raw image intensity with affine lighting model.	Direct image alignment.	Keyframe-based forward-compositional Lucas-Kanade image alignment with affine lighting model.
VI-DSO	Candidate points are chosen to be well spaced in image and have high gradient, activated points are successfully tracked and spread in 3D space.	Raw image intensity with affine lighting model.	Direct image alignment.	Keyframe-based forward-compositional Lucas-Kanade image alignment with affine lighting model, then associated with a pre-integrated IMU residual.
DSM	Candidate points are chosen to be well spaced in image and have high gradient, activated points are successfully tracked and spread in 3D space.	Raw image intensity with affine lighting model.	Direct image alignment.	Keyframe-based forward-compositional Lucas-Kanade image alignment with affine lighting model.
ORB-SLAM3	ORB feature detector.	ORB descriptor.	Hamming distance between descriptors.	Constant motion model leading to guided correspondence search, followed by bundle adjustment.
BASALT	FAST feature detector.	Image intensity patches for initial tracking and ORB descriptors for later refinement.	Kanade-Lucas-Tomasi feature tracking for image patches, Hamming distance on descriptor for ORB features.	Correspondences from feature tracker produce a reprojection error combined with IMU error terms that are jointly optimised, with initial estimate from the IMU.

Table 2.2: (Continued)

Method	Point Selection	Data Type	Data Association	Tracking
Kimera	Shi-Tomasi corner detector.	Image intensity patches.	Kanade-Lucas-Tomasi feature tracking.	Feature tracker finds correspondences between frames and across stereo pairs, which are used in 5- and 3-point RANSAC respectively. A term from the IMU rotation can also be incorporated.

Table 2.3: Comparison of the *mapping thread* concept in Keyframe SLAM for different systems.

Method	Local Mapping	Map Point Type	Local Optimisation	Keyframe Creation	Keyframe Culling
MonoSLAM	2D search for initialised features in a gated region based on predicted location.	3D	-	-	-
PTAM	Map initialised with 5-point stereo RANSAC, then map points found by fixed range epipolar searches.	3D	Windowed local bundle adjustment on keyframes.	Time since last keyframe as well as distance.	-
DTAM	A dense depth map is build through primal-dual optimisation of a photometric cost function with regularisation term.	Inverse depth	-	Visibility of pixels from previous keyframe in current frame.	-
KinectFusion	A depth frame with an associated pose is fused into the global 3D reconstruction with a volumetric truncated signed distance function.	3D	-	-	-
MSCKF	Mapped locations of tracked points are refined with a structure-only bundle adjustment.	Inverse depth	EKF update step based on finalised feature tracks.	All frames contribute a pose to the state vector.	When the keyframe limit is reached a third of the active frames are culled, equally spaced in time.

Table 2.3: (Continued)

Method	Local Mapping	Map Point Type	Local Optimisation	Keyframe Creation	Keyframe Culling
SLAM++	Candidate objects are fused into the global map using ICP alignment. New meshes are fitted to unexplained regions in the map with an active search.	3D Object	-	-	-
LSD-SLAM	A depth map is build using small-baseline SSD epipolar searches constrained by existing hypotheses.	Inverse depth	-	Distance and rotation limit since last keyframe.	-
SVO	Feature tracks are triangulated and added to global map once variance is sufficiently low.	3D	Pose-only bundle adjustment, then structure-only, then full local bundle adjustment.	3D map point visibility in current frame.	-
ORB-SLAM	Feature tracks are triangulated and refined in local bundle adjustment on all points in current keyframe.	3D	Local bundle adjustment on window of keyframes co-visible with current active keyframe.	Pixel visibility between keyframe and current frame, as well as number of frames since last keyframe.	Keyframes are culled based on redundant visibility of map points at the given scale or finer.
ElasticFusion	The depth map is fused with active portions of the global map during the joint photometric and geometric alignment which estimates the pose.	Surfel	Short-term local loop closures by attempting registration of active portion of map with inactive portions.	-	-

Table 2.3: (Continued)

Method	Local Mapping	Map Point Type	Local Optimisation	Keyframe Creation	Keyframe Culling
OKVIS	Triangulation of 2D-2D correspondences not yet associated with landmarks, both across stereo pairs and temporally with previous images.	3D	Local bundle adjustment with joint re-projection and IMU error terms.	Latest keyframe key-point visibility and global landmark visibility in current frame.	Oldest keyframe marginalised.
ROVIO	Successfully tracked features from direct alignment are kept in EKF state as long as they have good statistics.	Inverse depth	EKF update including intensity error of tracked image patches.	-	-
ORB-SLAM2	Feature tracks are triangulated and refined in local bundle adjustment on all points in current keyframe.	3D	Local bundle adjustment on window of keyframes co-visible with current active keyframe.	Pixel visibility between keyframe and current frame, as well as number of frames since last keyframe.	Keyframes are culled based on redundant visibility of map points at the given scale or finer.
DSO	A depth map is build using small-baseline SSD epipolar searches followed by fixed iterations of Gauss-Newton refinement.	Inverse depth	Windowed photometric bundle adjustment on active keyframes.	Field of view overlap with current keyframe, translation since last keyframe, and large changes in exposure time.	Hard limit on number of active keyframes, old keyframes marginalised based on visibility and keeping them well spread in 3D space.

Table 2.3: (Continued)

Method	Local Mapping	Map Point Type	Local Optimisation	Keyframe Creation	Keyframe Culling
VI-DSO	A depth map is build using small-baseline SSD epipolar searches followed by fixed iterations of Gauss-Newton refinement.	Inverse depth	Windowed joint photometric bundle adjustment, with IMU error term, on active keyframes.	Field of view overlap with current keyframe, translation since last keyframe, and large changes in exposure time.	Hard limit on number of active keyframes, old keyframes marginalised based on visibility and keeping them well spread in 3D space.
DSM	A depth map is build using small-baseline SSD epipolar searches followed by fixed iterations of Gauss-Newton refinement.	Inverse depth	Windowed photometric bundle adjustment on active keyframes.	Field of view overlap with current keyframe, translation since last keyframe, and large changes in exposure time.	Hard limit on number of active keyframes, keeping them spread in 3D space. Keyframes are also activated/deactivated based on co-visibility with current keyframe.
ORB-SLAM3	Feature tracks are triangulated and refined in local bundle adjustment on all points in current keyframe.	3D	Local bundle adjustment on window of keyframes co-visible with current active keyframe.	Pixel visibility between keyframe and current frame, as well as number of frames since last keyframe.	Keyframes are culled based on redundant visibility of map points at the given scale or finer.
BASALT	Points are mapped as a unit vector and inverse depth in the host keyframe they were first observed based on correspondences.	Inverse depth	Local bundle adjustment tightly coupled with IMU error term, done with respect to active keyframes and m recent frames.	Keyframes are created if a frame has landmarks initialised within it.	Older keyframes are marginalised as new frames become keyframes.

Table 2.3: (Continued)

Method	Local Mapping	Map Point Type	Local Optimisation	Keyframe Creation	Keyframe Culling
Kimera	Triangulation of feature tracks to 3D vertices that are used to generate a 3D mesh.	Mesh	Factor-graph solved at each keyframe.	Time since last keyframe and visibility of features.	-

Table 2.4: Comparison of the *loop closure and optimisation thread* concept in Keyframe SLAM for different systems.

Method	Relocalisation	Loop Closure	Global Optimisation
MonoSLAM	-	-	-
PTAM	-	-	Global bundle adjustment.
DTAM	-	-	-
KinectFusion	Manual user alignment of current depth map with the depth map of the most recent known pose.	-	-
MSCKF	-	-	-
SLAM++	A new local graph is created when tracking is lost, then matched to the global map once it is large enough.	Loops are detected by matching fragments of the global graph with the full graph in the same process as relocalisation.	Pose graph optimisation.
LSD-SLAM	-	OpenFABMAP appearance-based matching of keyframes in pose graph.	Pose graph optimisation.
SVO	-	-	-
ORB-SLAM	DBoW2 appearance-based matching of current frame to keyframes followed by RANSAC pose estimation.	DBoW2 appearance-based matching of current keyframe to past keyframes followed by RANSAC pose estimation of loop closure and updated pose graph optimisation.	Pose graph optimisation.
ElasticFusion	-	Global loop closure by appearance-based lookup of randomised fern encoding database, local loop closure by registering active map with inactive map.	Optimisation of a deformation graph.
OKVIS	-	-	-
ROVIO	-	-	-

Table 2.4: (Continued)

Method	Relocalisation	Loop Closure	Global Optimisation
ORB-SLAM2	DBoW2 appearance-based matching of current frame to keyframes followed by RANSAC pose estimation.	DBoW2 appearance-based matching of current keyframe to past keyframes followed by RANSAC pose estimation of loop closure and updated pose graph optimisation.	Pose graph optimisation as well as global bundle adjustment which is aborted and restarted if a new loop closure makes it invalid.
DSO	-	-	-
VI-DSO	-	-	-
DSM	-	Old keyframes are reactivated based on estimated co-visibility with the active keyframe as well as how well the candidate inactive keyframe ‘fills in the gaps’ in the projects of active keyframes into the latest keyframe.	-
ORB-SLAM3	DBoW2 appearance-based matching of current frame to keyframes followed by RANSAC pose estimation.	DBoW2 appearance-based matching of current keyframe to past keyframes followed by RANSAC pose estimation of loop closure and updated pose graph optimisation.	Pose graph optimisation as well as global bundle adjustment which is aborted and restarted if a new loop closure makes it invalid.
BASALT	-	A separate system of detecting and matching ORB features is used to add loop-closure constraints between keyframes.	Global bundle adjustment is performed on the separately detected and matched ORB features, with the inclusion of an addition nonlinear factor recovery term.
Kimera	-	DBoW2 appearance-based matching of current keyframe to past keyframes followed by pose graph optimisation.	Pose graph optimisation.

2.5 Distributed calibration, localisation, and SLAM

The foundation of distributed robotic vision lies in *distributed computing and consensus*. Distributed computer vision is generally achieved through the use of methods such as average-consensus, belief propagation, and distributed optimisation. These algorithms involve viewing the network as a graphical model where cameras are nodes and their relationships to other cameras are edges. Whilst research into these algorithms has often only considered static graphs, there has been significant work in investigating the effects of switching topologies, time-delays, and non-ideal edges. The motivation of this section is exploring how the algorithms of the previous sections can be adapted to distributed networks, filling in the third component identified in Figure 2.1.

Section 2.5.1 first looks at the use of graphical representations of camera sensor networks through the *vision* graphs and *communication* graphs, with a range of useful properties of the graph defined. The problem of determining the topology of these graphical representations is also explored. Then in Section 2.5.2, research is discussed dealing with the distributed algorithms of average-consensus, belief propagation, and distributed optimisation, looking at the foundational work in each area as well as the wide range of applications each type of algorithm has seen.

Applying this knowledge, Section 2.5.3 looks at the use of the graphical CSN framework discussed in Section 2.5.1 and the distributed algorithms discussed in Section 2.5.2 to perform multi-view calibration and localisation algorithms in distributed camera sensor networks, looking at how these distributed algorithms are applied to the data association, linear systems, and non-linear refinements of the various problems. Section 2.5.4 then looks at how these methods are applied to distributed multi-robot SLAM, discussing which parts of the SLAM pipeline have been adapted to distributed processing and in which ways. Finally, Section 2.5.5 summarises these findings and the open problems that have been identified.

2.5.1 Distributed camera sensor networks

Applications of calibration, localisation, VO, and SLAM to distributed CSNs are most often formulated as collections of consensus problems on undirected graphs. This is where the cameras are modelled as nodes and connections between them are modelled as edges. Radke, Tron and Vidal [3, 8, 33], Devarajan et. al. [7, 156–158], and Song et.al [1] take the approach of modelling distributed CSNs as two separate undirected graphs, a *communication* graph, $\mathcal{G}_C = \{\mathcal{V}, \mathcal{E}_C\}$, and a *vision* graph, $\mathcal{G}_V = \{\mathcal{V}, \mathcal{E}_V\}$. For both of these graphs $\mathcal{V} = \{1, \dots, N\}$ is the set of all camera sensor nodes, $\mathcal{E}_C \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges containing $(i, j) \in \mathcal{E}_C$ pairs of nodes that are able to communicate directly, and $\mathcal{E}_V \subseteq \mathcal{V} \times \mathcal{V}$ is the set of

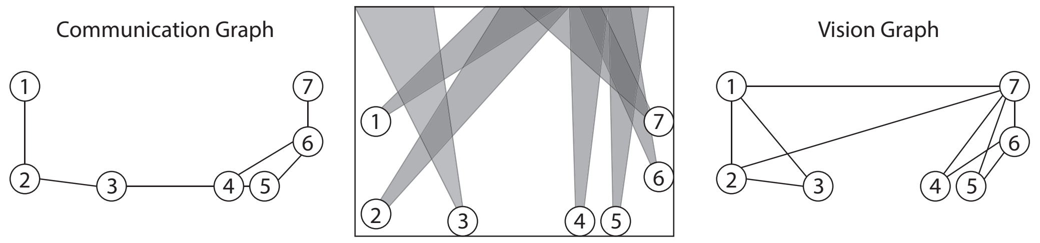


Figure 2.24: Comparison of the vision graph (left) and communication graph (right) for a CSN (centre).

edges containing $(i,j) \in \mathcal{E}_V$ pairs of camera nodes with overlapping fields of view. Additional useful properties of these graphs are the set of neighbours of a node, $\mathcal{N}_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$ for either $\mathcal{E}_C$ or $\mathcal{E}_V$, the degree of a node, $d_i = |\mathcal{N}_i|$, and the maximum degree of a graph, $\Delta_\mathcal{G} = \max_{i=1,\dots,N}\{d_i\}$.

The vision graph and communication graph can be significantly different, as can be seen in Figure 2.24 where the communication graph is determined by physical distance in this simplified representation and the vision graph is determined by sufficient overlap of field-of-view. Most computer vision algorithms are designed based on the vision graph whilst leaving consideration of the communication graph for implementation. Radke distinguishes purely *distributed algorithms* as those which only rely on local decisions based on information from their immediate neighbours, with the primary design goal being to approach the results obtained by fully centralised algorithms [3].

The first step in most distributed computer vision applications is the determination of the vision graph – a process referred to as *topology estimation.* This comes in two types; *overlapping* vision graphs where the majority of cameras share a significant portion of the field-of-view with neighbours and *non-overlapping* where the cameras do not share fields-of-view. This problem is very similar to the multi-camera localisation problem discussed in Section 2.3.3, however, instead of estimating rigid-body transformations between cameras the goal is to infer edges between camera nodes on the graph if sufficient overlap of field-of-view exists or, in the case of non-overlapping cameras, if the cameras have related entry and exit points.

Techniques for non-overlapping estimation are generally based around tracking moving targets whose relative visibility is used to infer the edges. An early implementation of this was done by Makris et al. [159], where entry and exit points for a camera's field-of-view are automatically learnt and then edges between nodes on the vision graph are inferred as targets move between views. Rather than finding correspondences between specific targets, they instead match the rates of targets leaving one view and entering another and use a probabilistic model to infer the relationships between the 'blind' spots between cameras. Marinakis and Dudek [160] used a similar approach with a non-discriminative model to infer the topology through

Monte Carlo expectation maximisation where they model the non-overlapping scene with nodes for the cameras as well as additional nodes representing sources/sinks from and to which agents travel in the environment outside the field of view. They also include a delay model that accounts for the differing number of agents in the scene as well as agents that take different lengths of time to travel between views. Zou et al. [161] also developed a method based on entry and exit between views, but used facial recognition to track targets across camera nodes rather than previous techniques which relied on known targets. They argue that using overall appearance of a target can have problems if different targets are wearing similar clothes or if appearance changes due to different lighting, whilst facial appearance should be more discriminative and consistent. Farrell et al. [162] developed an interesting higher-order Bayesian framework that not only learnt the adjacency, but also the likelihood that a target will traverse a particular edge. Whilst the earlier discussed methods were applied to networks of 6-9 cameras, this method was aimed at medium sized networks of 10-1000 cameras. Farrell and Davis [163] later improved upon these tracking-based methods using an information-theoretic appearance model to weight the targets based on distinctiveness. These non-overlapping methods are generally designed for stationary CSNs, such as surveillance networks, where it is reasonable to rely on large numbers of moving targets throughout the scene. Although robotic vision applications can have regions and periods of no overlap, the cameras are usually not stationary and can only rely on the presence of moving targets in specific applications.

For overlapping estimation there is a wider variety of approaches taken including tracking, object detection and feature point matching. Kulkarni et al. [164] used object detection with time-stamped images of a reference object being compared to determine regions of overlap. Instead of assuming that there are moving targets in the scene, as is the case for most non-overlapping methods, the authors manually insert a reference object into the scene at random locations which can be observed simultaneously in overlapping networks. Although simple for their small CSN, this method quickly becomes complicated as the network grows in size. On the other hand, this method can be done at the same time as multi-camera calibration and localisation if the reference object can act as a calibration object. Alternatively, Avidan et al. [165] proposed a method based on wide-baseline stereo matching of feature points between views to directly determine edges. They initially look at this from a centralised perspective where all comparisons are made at a single processor that has access to all information, however, they expand this to a distributed implementation where feature points are only shared along edges of the communication graph. In this case, if a node determines that two of its neighbours can see the same points it can share that knowledge which allows edges that do not appear on the communication graph to be found on the vision graph. Cheng et al. [166] employed a similar approach with what they refer to as 'feature digests'. A fixed number of feature points are selected from each image, based on distinctiveness and spatial distribution, and compressed using principal component analysis. These feature digests are broadcast throughout the network for stereo matching to form edges on the vision graph. Unlike Avidan, Cheng's method requires the feature digest to be sent to every node and therefore isn't highly scalable. However, feature digests are a novel approach to reducing the communication load, and a combination of the two algorithms would be interesting to see. Van den Hengel et al. [167] proposed a contrasting estimation method which they refer to as *exclusion*. Rather than building up evidence for an edge, they instead assumed that the vision graph is fully connected and used contradictory evidence to prune edges, homing in on the correct topology. They argue that this process requires much less data than building up evidence, however, this assumption is specific to the expected connectivity of the network.

There has also been a number of methods for overlapping networks that utilise motion in similar manners to the non-overlapping case. Mandel et al. [168] and Ermis et al. [169] both used coherent motion between views to form vision graph edges. The method of Mandel et al. assumes that all camera nodes can communicate with all others, however, with sufficient routing along the communication graph this is not a necessity. Their method is based on coherent motion of foreground regions, with overlap inferred from a probabilistic analysis of simultaneous activity in image regions. Their method does not utilise specific mobile targets moving between views like the non-overlapping method, but simply considers the occurrence of motion as an event that is compared to other views. The method of Ermis et al. uses activify features which they define as the occupancy duration of foreground pixels. Their method works well in networks where the cameras have significantly different views as the corresponding pixels should still be occupied for a similar duration despite the direction from which the object is observed. Also similar to non-overlapping methods, Esterle et al. [170] produced a vision graph as a by-product of tracking in which they used a socio-economic model where camera nodes buy and sell ownership of the targets and use an auction system to maximise their utility. The intention of their method is to manage responsibility of tracking targets, however, the vision graph is built based on the 'market activity' by adding links when a trade occurs. This method of vision graph formation is novel as one of very few topology estimation algorithms with robustness to changing topologies.

As can be seen, the vast majority of methods discussed are all built around networks that are static at the time of topology estimation and often require one or many moving targets throughout the scene. Alternative to the moving target, algorithms such as that suggested by Kulkarni et al. [164] instead use a manually placed reference object and others still rely on static scene constraints. These kinds of algorithms pair well with the calibration and localisation task which is also generally performed with static positions. If using a calibration method that utilises a calibration object then a method like Kulkarni et al. [164] can be used, whereas if self-calibration is being performed then methods such as those of Avidan et al. [165] or Cheng et al. [166] could be used. In a mobile robotic vision application, such as VO or SLAM, this vision graph formation could be seen as a continuous process that finds and updates the vision graph as motion changes the field of view overlap.

2.5.2 Distributed consensus algorithms

Utilising these graphical representations of distributed CSNs, robotic vision problems such as calibration, localisation, VO, and SLAM are then implemented using consensus algorithms that operate on these graphs. Three such types of distributed consensus algorithms are average consensus, belief propagation, and distributed optimisation. Average consensus allows each node to compute the average of estimates across a number of nodes using a control law that is based only on the exchange of information between direct neighbours. This simple algorithm can be extended to more complicated cases that allows the solving of linear and even nonlinear systems. Belief propagation is a similar class of algorithm that involves message passing between direct neighbours and is used to perform inference for probabilistic problems on graphs. Finally, distributed optimisation is a group of methods that allow the objective function of an optimisation problem to be split according to what information is available locally at each node under simple communication protocols.

Average consensus

One of the most common types of distributed algorithm is that of *average consensus*, which achieves consensus by utilising only local computations and data sharing with direct neighbours. Consider a network where each node has a local measurement or estimate $x_i \in \mathbb{R}$ relating to the same quantity and the desired consensus value is the average of these measurements given by Equation 2.15. Average consensus computes this value based on the fact that the equilibrium point of Equation 2.16 is equal to Equation 2.15, with $z_i(t)$ being the state of node i at time t and $\epsilon \leq \Delta_{\mathcal{G}}^{-1}$ is the step size. At each time step, the current value at a node is augmented by its ‘disagreement’ with neighbouring nodes. It was demonstrated by Olfati-Saber and Murray [171] that after each update average of the values across all nodes remains unchanged, however, the disagreement at each node and the global disagreement are reduced. Therefore, each node will converge towards the global average value. An

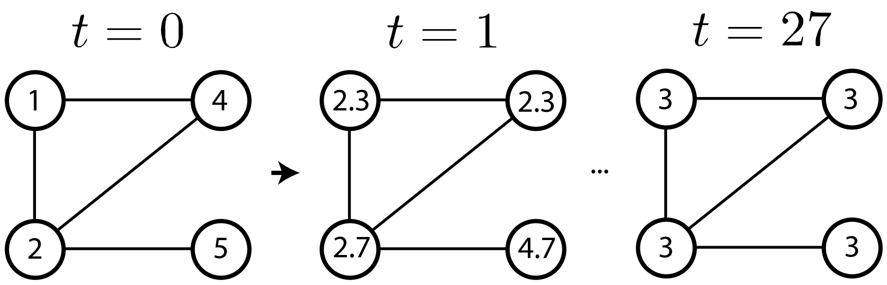


Figure 2.25: An example network performing average consensus with four nodes and $\varepsilon = 1/3$. The initial state of the system at time $t = 0$ is shown on the left. At each time-step, the nodes change their value based on their disagreement with their neighbours according to Equation 2.16. At each time-step, the global average value across all nodes remains the same. Each node converges to this average value.

example of average consensus is shown in Figure 2.25 for a network of four nodes and $\varepsilon = 1/3$. In this case, it takes 27 iterations until the nodes converge on the average value to four significant figures.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.15}$$

$$z_i(t+1) = z_i(t) + \varepsilon \sum_{j \in \mathcal{N}_i} (z_j(t) - z_i(t)), \quad z_i(0) = x_i \tag{2.16}$$

This algorithm can be trivially extended to multivariate data $x_i \in \mathbb{R}^n$, as well as used to perform distributed linear algebra. Tron and Vidal [2] show that in a system where each node has a local measurement matrix $\mathbf{A}_i \in \mathbb{R}^{n_i \times m}$ and measurement vector $\mathbf{b}_i \in \mathbb{R}^{n_i}$, one can solve linear least squares problems $\mathbf{Ax} = \mathbf{b}$ for the global measurement matrix $\mathbf{A} = [\mathbf{A}_1^T, \ldots, \mathbf{A}_N^T]^T$ and global measurement vector $\mathbf{b} = [\mathbf{b}_1^T, \ldots, \mathbf{b}_N^T]^T$ using Equations 2.17 and 2.18. This, in turn, can be solved using the multivariate form of Equation 2.16. They also demonstrate a similar approach for calculating singular value decomposition and nullspace estimation.

$$\frac{1}{N}\mathbf{A}^T\mathbf{A} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{A}_i^T\mathbf{A}_i \tag{2.17}$$

$$\frac{1}{N}\mathbf{A}^T\mathbf{b} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{A}_i^T\mathbf{b}_i \tag{2.18}$$

There has been significant investigation into the stability of these consensus algorithms in networks of switching topologies and time-delays in communication, which is highly important for robotic vision applications. Early work by Olfati-Saber et al. [35, 171] addressed these two problems separately. For switching topologies, it was found that on undirected graphs for any time t_k before convergence, average-consensus could handle occasionally disconnected topologies as long as the graph became fully connected again at some point in $[t_k, \infty)$. For time-delays, they found that the convergence occurred as long as the delays remained under an upper bound inversely proportional to the algebraic connectivity of the graph. That is, the fewer the edges on the graph, the more robust it was to the effects of time-delays. Ren and Beard [172] found that convergence could be achieved on switching topologies as long as the union of all the directed interaction graphs contained a spanning-tree. Rabbat et al. [173] designed an improved average-consensus algorithm that converged faster on switching topologies than previous algorithms by implementing a so-called 'fair' loss function. Xiao and Wang, Sun et al., and Lin and Jia all analysed convergence on systems that have both switching topologies and time-varying time-delays [174–176]. Using Lyapunov stability analysis and linear matrix inequality methods, they found that the convergence occurred as long as the union of graphs along a bounded period of time contained a spanning-tree. Chen et al. [177] developed a variation of average-consensus called *corrective consensus* which was more robust to switching topologies and packet losses by recording the rate of change at each iteration and broadcasting this during "corrective iterations". There has also been work on leader-based consensus tracking, where average-consensus is used to make a switching distributed network converge on a leader node's changing state. Su and Huang [178], and Chen et al. [179] found that tracking could be performed on fully connected switching networks if the control gains were properly chosen. Xu et al. [180] had similar results to the two previous, but noted that performance could be improved by additionally tracking the estimated tracking error of each node's neighbours. Wen et al. [181] also noted the importance of dwell time, however their design didn't require all graphs in the switching topology to be fully connected, given sufficient communication rate.

There has also been work on some novel changes to the basic average consensus algorithm, such as in the work of Seyboth et al. [182] which presented an interesting 'event-triggered' average consensus where broadcasts are only sent to neighbours once the current local estimate and last broadcast value become sufficiently different, greatly reducing communication costs. Nowzari and Cortes, and Li et al. improved this design by removing any requirement for global knowledge of algebraic connectivity of the network [183, 184]. Zuo and Tie [185] increased robustness to external disturbances through a non-linear analysis, and Hadjicostis et al. [186] improved robustness to packet loss through the exchange of running sums which helped detect discrepancies between nodes.

One of the earliest applications of average consensus was on the problem of flocking, where a number of mobile agents align their heading directions. Jadbabaie et al. [187] considered the model of Vicsec et al. [188] with a network of mobile agents that were modelled as points or particles on a plane each with the same speed but different headings. They then considered the neighbours of an agent to be those within a certain radius and have each agent update its heading to be the average of itself and its neighbours. The authors demonstrate the convergence occurs despite the network's changing topology. In their formulation of flocking, the specific value to which the nodes converge is not the average of the initial state but instead dependent on the edges as the network evolves. They do demonstrate 'leader following', however, where one node can retain its value which will force all other nodes to converge to it. This does not work in the case where more than one node attempts to be the leader and is not a truly distributed algorithm. Savkin [189] analysed a discrete version of this model and showed that all nodes will converge to a value and if the network remains connected, then all nodes will have converged to the same value. Olfati-Saber [29] presented a number of flocking algorithms that aimed to adhere to three flocking rules of cohesion to a flocking centre, separation between nodes, and alignment of velocity. Their first algorithm met all three goals but only remained as a single connected graph in a narrow set of initial states, whilst their second algorithm solved this issue by introducing a 'group objective' term that has similar effect to the leader following of Jadbabaie et al. Their final algorithm introduced a third term for obstacle avoidance where obstacles are modelled as an additional type of agent. Their work showed that this group objective is important to prevent the network from fragmenting into several smaller graphs with different consensus values. This is not general to all average consensus problems, as it is due to the fact that the connectivity of the graph is determined by the states that are attempting to converge.

Further and more advanced robotics applications of average consensus include formation control, coverage control, and task assignment. Formation control is similar to flocking, particularly when compared to the flocking with a group objective shown by Olfati-Saber [29]. Lin et al. [190] first considered a protocol where nodes communicate with only the next node 'above' them according to assigned numbers. They showed that since this protocol could converge to a point in the absence of collision, then it could also be used for formation control with added displacement goals at each node. They then expanded this to the model more common in flocking where nodes had edges based on proximity and demonstrated that the same was true. One difference to the flocking algorithms discussed above is that rather than each node approaching the average of all its neighbours, the authors used the disagreement between nodes like in Equation 2.16. This allowed them to know that the value converged upon was the centroid of the starting values. Muhammad and Egerstedt [191] also analysed the range-based connectivity model of flocking for consideration in formations, demonstrating that a number of geometric properties could be inferred from the connections of the graph without actually knowing the positions of individual nodes. Cortes et al. [192] consider a very similar problem to formation control, designing a distributed algorithm for multi-vehicle task assignment that ensures optimal coverage for a remote sensing task. The authors consider dividing a space up using Voronoi diagrams based on the locations of mobile robots. They then present distributed gradient descent algorithms that seek uniform distribution of the robots across the environment.

There has also been work on general approaches to sensor fusion and filtering, which has significant applications in robotic vision and SLAM. Xiao et al. [31] demonstrated that average consensus could be used to implement maximum likelihood sensor fusion for linear systems by expanding Equations 2.17 and 2.18 to incorporate covariance. Like with other linear systems solvable by average consensus, this requires that all estimates and covariances are separable between nodes, however, in such a situation the problem is able to converge on networks of changing topology as long as the union of all infinitely occurring graphs is connected, even if individual graphs are not. They demonstrate this under two weighting schemes — maximum degree weights, which require global knowledge of the number of nodes in the system, and metropolis weights, which do not. The ability of the latter to converge is important because the upper bound on step size $\epsilon \leq \Delta_{\mathcal{G}}^{-1}$ given for Equation 2.16 does technically rely on global information, whereas the metropolis weights do not. Olfati-Saber and Shamma [193] expanded the use of average consensus into the domain of tracking changing signals with their introduction of *consensus filters*. They had each node track the signal by minimising both the disagreement between nodes on the current value as well as the disagreement between the current value and new measurements, effectively acting as a low-pass filter. They were also able to implement a similar control law for a high-pass filter and therefore a band-pass filter. This was used by Spanos et al. and Olfati-Saber to introduce Kalman Consensus Filters (KCF), which use 'micro-Kalman Filters' at each node and bring local beliefs into global consensus using the average-consensus low-pass and band-pass filters [194, 195]. Their first implementations performed consensus on the measurements and covariances used as inputs to the state estimation of the update stage meaning that each node would calculate the same estimates, however, Olfati-Saber later demonstrated that superior performance was seen by instead performing consensus afterwards on the differing state estimates [196]. In analysing these KCFs, Olfati-Saber found that the optimal form of the problem was not scalable in the number of nodes due to correct covariance propagation ideally requiring all-to-all connections, however, this was solved by deriving an approximation to the covariance that is scalable in the number of nodes and was demonstrated to be a stable alternative.

These KCFs and general approaches to sensor fusion and filtering are highly applicable to robotic vision as many of the fundamental implementations of SLAM, such as MonoSLAM [88] or MSCKF [91], utilise extended Kalman filters and, as was discussed in Section 2.4.2, VO and SLAM are often interpreted as filtering problems.

Average consensus is one of the most fundamental algorithms for distributed computing and is used to some degree in almost every distributed computer vision application. These consensus algorithms are extremely useful in solving a wide range of optimisation problems on systems that have limited communication abilities. By only operating on the disagreement between an estimate at a node and the estimates at neighbouring nodes, global averages can be determined which can then be used to perform a wide range of operations including flocking, formation control, linear algebra, and filtering. All of these operations have direct applications to robotic vision tasks, such as the linear systems in calibration and the filtering often used in SLAM.

Belief propagation

Another consensus algorithm is the message-passing algorithm of *belief propagation* (BP), first proposed by Pearl for performing inference on probabilistic graphical models [197]. The original form was developed for operation on directed graph representations of inference problems, however, the generalised form functions on a range of graph types including factor graphs and Markov random fields (undirected graphs). Applying the problem to the undirected graphical structures discussed in this chapter, each node has a measurement z_i as is the case with average consensus. These are interpreted as random variables with node distributions $\phi_i(z_i)$ and edge distributions between nodes i and j as $\psi_{ij}(z_i, z_j)$. At each iteration, nodes send messages $m^{(t)}_{i\to j}(z_j)$ to neighbours regarding knowledge of that node's variable(s) comprising of the 'sum-product' of the potentials with messages from the previous iteration (excluding the message from the receiving node), shown in Equation 2.19 where $\mathcal{N}_i \setminus j$ means the neighbours of node i excluding node j. This process is demonstrated visually in Figure 2.26 showing how the receiving node does not have its previous messages incorporated into the message sent to it. The node's belief about its own variable $b^{(t)}_k(z_i)$ is the product of the node potential and all incoming messages from that time instant, shown in Equation 2.20. Pearl showed that on acyclic graphs, such as spanning trees, the beliefs were guaranteed to converge to the exact marginal value. However, loopy graphs don't always converge and only do so to approximate values.

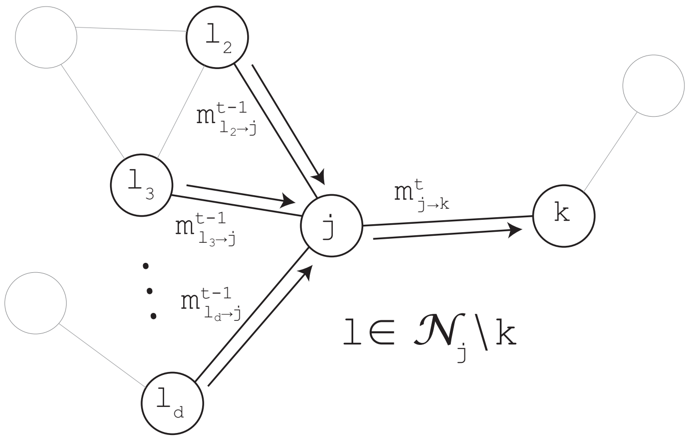


Figure 2.26: Message propagation in belief propagation, where the message sent to a target node is comprised of the earlier messages received at the sending node, except that received from the target node.

$$m^{(t)}_{i\to j}(z_j) \propto \int_{z_i} \phi_i(z_i)\psi_{ij}(z_i, z_j) \prod_{k\in\mathcal{N}_i\setminus j} m^{(t-1)}_{k\to i}(z_i)dz_i \tag{2.19}$$

$$b^{(t)}_k(z_i) \propto \phi_i(z_i) \prod_{k\in\mathcal{N}_i} m^{(t)}_{k\to i}(z_i) \tag{2.20}$$

Murphey et al. [198] showed that loopy BP can converge for a number of cyclic graph structures, looking at a range of graphical structures common to various problems across different fields, include image processing. They found that whilst oscillations did occur on a number of structures, accurate results were seen on graphs of widely varying sizes with differing amounts and sizes of loops. Yedidia et al. [199] reformulated a *generalised belief propagation* which had greater performance on loopy graphs, demonstrated on regular lattice graphs. Their method formulates the messages as between regions of the graph rather than just on individual nodes. Of particular interest to the distributed processing considered in this chapter, Weiss and Freeman [200] extended BP to work with continuous-variables with Gaussian distributions. Their *Gaussian belief propagation* (GaBP) was quite simple in that each message only comprised of two scalar values for the mean and variance. This form of the algorithm is capable of computing maximum likelihood and maximum a posteriori estimation, however, the algorithm is still only approximate on loopy graphs. They provide a number of sufficient conditions for GaBP to produce exact means on loopy graphs and find, similar to results with other forms of BP, that faster convergence results in more accurate variance. Relating this algorithm to average consensus, Moallemi and Van Roy [201] considered the situation described above where each node had a single scalar measurement with the goal of determining the average using GaBP. They demonstrated that convergence can be improved by introducing an attenuation factor that prevents the variance estimates from going out of control. This attenuation factor affects both the convergence speed as well as the final accuracy. They explored how this parameter can be chosen based on the graph structure with goals of convergence to within a defined accuracy. Using their results, the range of distributed algorithms discussed relating to average consensus could be equivalently implemented with GaBP. Olfati-Saber et al. [202] also explored the relationship between BP and average consensus by re-deriving the BP messages and beliefs from the average consensus control law, relating the necessary attenuation factor to the step size ε in Equation 2.16. Alternatively, Ihler et al. proposed *non-parametric belief propagation* (NBP) [203], a particle-based method with arbitrary distributions being represented by mixtures of Gaussians. This required more memory than GaBP, but can represent non-Gaussian distributions.

A number of authors have explored the convergence properties of GaBP and NBP in order to find improvements for the approximate inference on loopy graphs as well as finding situations that can provide exact inference. Johnson, Malioutov and Willsky [204, 205] provided a framework for analysing GaBP based on 'walks on a graph' in Gaussian graphical models, with the correlation between variables being related to the sum of all walks between the two variables. This allowed them to derive a condition for convergence based on the spectral radius of the matrix describing the graph. On the convergence of GaBP, Su and Wu, and Du et al. [206–209] found that convergence was exponential for positive semi-definite information matrices at initialisation. These conditions, however, only describe convergence of the Gaussian mean whereas the variance is not guaranteed convergence in these conditions. Exploring the speed of convergence for NBP, Sudderth et al. [210] demonstrated that performance could be improved for the marginalisation of its non-Gaussian distributions through importance-weighted sampling of the message particles to ensure that processing power is focused on the most probable regions of the distribution. Savic et al. [211] also improved the performance of NBP using a 'tree-reweighted' method, where the graph edges were weighted by the frequency that the edge appears the set of all spanning-trees for the graph's topology. This method of improving convergence speed and accuracy through message weighting is also applicable to GaBP, similar to the attenuation factor used by Moallemi and Van Roy [201].

These belief propagation algorithms can be applied to a wide range of problems — general BP to discrete probabilistic problems, GaBP to problems with continuous random variables that can be modelled as Gaussian, and NBP to continuous random variables of more arbitrary distributions. Kschischang et al. [94] showed that the 'sum-product' quality of BP related it to a wide range of similarly arranged problems including fast Fourier transform and Kalman filtering. They demonstrated that such problems were equivalent across their contexts and could be interpreted as factor graphs, therefore implementing them efficiently using BP, however, they did not demonstrate such problems being solved in a distributed manner. There have, on the other hand, been a range of applications directly to distributed sensor networks. Olfati-Saber et al. [202] reinterpreted the belief propagation algorithm as a type of average consensus, demonstrating how a distributed network of sensors can be employed for distributed hypothesis testing. The relationship between average consensus and the various forms of belief propagation demonstrates the parallels between algebraic problems and probabilistic problems. Utilising NBP, Ihler et al. [203] demonstrated how the localisation problem could be solved where an array of sensor nodes in a 2D environment have noisy measurements of the distances between certain nodes, as well as where some nodes have noisy absolute measurements of their own position. Modelling the noise as mixtures of Gaussians, they formatted the maximum a posteriori (MAP) estimation of all positions as an inference problem that could be solved with message passing. Since NBP messages are collections of numerous particles, as in standard particle filtering, sampling techniques were required to combine the messages and produce new collections of particles at each stage. In their example simulations, they demonstrated that the NBP method was close in accuracy to the centralised MAP estimate with the advantage of being much more computationally efficient on larger networks of around 100 sensor nodes.

There has been extensive research into GaBP and its application to distributed networks. Shental et al. [212] demonstrated how GaBP could be used to solve linear systems of equations of the form $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a symmetrical matrix. By building a undirected graph where the edges are represented by the off-diagonal entries in $\mathbf{A}$, GaBP was employed to solve for $\mathbf{A}^{-1}\mathbf{b}$ without the need for matrix inversion. This is different to the average consensus method of solving linear systems of equations by finding $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ at each node then performing the matrix inversion locally at each node. In the average consensus case, each node had a 'slice' of the problem, whereas in this case, the graph structure is described by $\mathbf{A}$. They were able to show that the Gaussian distributions converged to exact means, although approximate variances, when $\mathbf{A}$ was diagonally dominant. As was the case discussed with average consensus, the ability to solve linear systems of equations in a distributed manner opens up the possibility to solve a wide range of distributed problems. Bickson et al. [213] used this process to implement a minimum mean-squared error (MMSE) linear detector for the multiuser problem, and later demonstrated that a distributed Kalman filter could be implemented using two passes of their MMSE estimator [214]. In both of these problems, the symmetrical information matrix was interpreted as the structure of the undirected graph. Bickson et al. [213] also followed a similar approach to implement a distributed *support vector* *machine* for classification and regression. These three examples of GaBP-based distributed algorithms were demonstrated as methods for efficiently computing the solutions in a centralised system, separating problems for parallel processing, as well as solving the problems in networks of distributed processing nodes.

This wide range of research has made belief propagation an extremely well understood distributed algorithm with a versatile range of forms useful for many types of problems. As will be discussed in Section 2.5.1, belief propagation has found important use in a number of distributed computer vision applications. Similar to average consensus, the ability to calculate linear systems of equations in a distributed manner through only local information and messages with direct neighbours opens up a range of possibilities. In fact, belief propagation provides an interesting probabilistic interpretation to the many applications of average consensus.

Distributed optimisation

There are a range of methods for solving optimisation problems in distributed systems where the global objective function can be separated into a number of local objective functions. Consider the objective given in Equation 2.21 where each node i in a system has some local objective f_i and variable $\mathbf{x}_i \in \mathbb{R}^n$. The goal of the optimisation is to jointly minimise all local objective functions whilst maintaining the constraint that all nodes agree. That is, all local $\mathbf{x}_i$ are equal to some variable $\mathbf{z}$. In a distributed system, it is clearly desirable to have each node optimise its local objective locally and use a consensus algorithm to ensure that all node reach and agree on the global minimum.

$$\begin{aligned} \text{minimise} \quad & f_1(\mathbf{x}_1) + \ldots + f_N(\mathbf{x}_N) \\ \text{s.t.} \quad & \mathbf{x}_i - \mathbf{z} = 0 \end{aligned} \tag{2.21}$$

This problem is often represented using an *augmented* Lagrangian function that includes an augmentation factor that enforces the constraint further [215]. The augmented Lagrangian for Equation 2.21 is given in Equation 2.22, where the first term of the sum is the local objective function, the second term involves the Lagrangian multiplier $\mathbf{y}$, and the third term is the augmentation factor that enforces the agreement of each local variable to the global consensus value. As can be seen, just as the local objectives are separable across the nodes, so too is the augmented Lagrangian. Not all distributed optimisation methods utilise the augmentation factor, such as in dual decomposition, however, there are many similar and often equivalent *proximal methods* that do include this factor which affords greater robustness to the

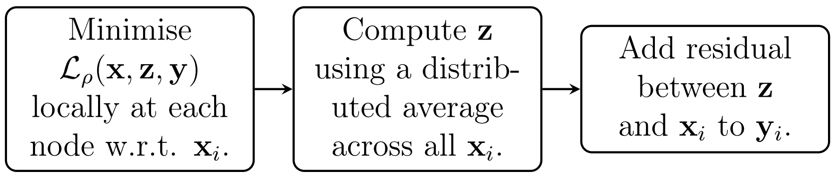

Minimise $\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y})$ locally at each node w.r.t. $\mathbf{x}_i$.
Compute $\mathbf{z}$ using a distributed average across all $\mathbf{x}_i$.
Add residual between $\mathbf{z}$ and $\mathbf{x}_i$ to $\mathbf{y}_i$.

Figure 2.27: One iteration of distributed optimisation with ADMM using the Lagrangian function given in Equation 2.22. Each iteration consists of a local optimisation, a distributed average, and a local addition.

algorithms [216].

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i) + \mathbf{y}_i^T(\mathbf{x}_i - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x}_i - \mathbf{z}\|_2^2 \tag{2.22}$$

The Lagrangian given in Equation 2.22 is given in the form utilised by one such proximal method, *alternating direction method of multipliers* (ADMM) [215]. In ADMM, the problem is solved in three stages: firstly the Lagrangian is minimised for $\mathbf{x}$, then for $\mathbf{z}$, and finally an update is performed for $\mathbf{y}$. In this algorithm, $\mathbf{z}$ can be interpreted as enforcing the consensus constraint and $\mathbf{y}$ measures the cumulative cost of disagreement. In fact, the $\mathbf{z}$-update step reduces to a distributed average over all $\mathbf{x}_i$ and the $\mathbf{y}$-update step simply involves adding the current residual between $\mathbf{x}_i$ and $\mathbf{z}$ to the current $\mathbf{y}_i$. Therefore, the algorithm is done using a local optimisation, a distributed average, and a local addition, as shown for one iteration in Figure 2.27, producing an entirely distributed algorithm for appropriate objectives. Although this description has been given specifically for ADMM, it can be shown that this holds close relation to other methods such as forward-backward splitting, Dogulas-Rachford splitting, Split Bregman and alternating Split Bregman algorithms, and a variety of other proximal splitting methods [217, 218].

These proximal splitting methods have been applied to a range of problems in a similar manner to average consensus to produce a number of distributed estimators. Rabbat et al. [173] considered how dual decomposition, a similar decomposition method that excludes the augmented factor, can be used to solve general optimisation problems that are split over the nodes in an undirected graph where links between nodes were unreliable. They also demonstrated that average consensus could be derived from their framework in the case where the objective is to minimise the sum of squares and demonstrated the resilience of this average consensus to link failures in the network. Schizas et al. [219] used a similar approach with ADMM to develop a range of robust estimators for deterministic signals, assuming that each node in an undirected graph has a measurement vector and that the graph is fully connected. They considered the problem of distributed estimation with designs for maximum likelihood estimators (MLE), in systems with known probability density functions, and best linear unbiased estimators (BLUE), for systems with known linear measurement models. Rather than have all nodes estimate the consensus variable $\mathbf{z}$, they only estimated it at 'bridge nodes' which were selected such that all regular nodes have a bridge node as a neighbour and all nodes that are separated by one hop share a common bridge node neighbour. This reduced the amount of communication required whilst still allowing the consensus variable to be calculated. They demonstrated that their estimators were resilient to communication noise as well as quantisation error. In further work, Schizas et al. [220] expanded their approach to random signals as well as non-stationary signals. They presented both maximum *a posteriori* (MAP) estimator and a linear minimum mean squared error (LMMSE) estimator. For the distributed MAP estimator, it was derived in the same manner as their ML estimator with the main difference being the inclusion of a prior. They also use the same methodology to develop a distributed Kalman filter and smoother for tracking a non-stationary signal. Compared to the average consensus-based methods of Olfati-Saber [195] and Spanos et al. [194], their method is capable of tracking fast-changing signals.

Proximal methods have also been applied to computer vision applications for both the ability to produce distributed algorithms as well as their excellent convergences and performance qualities. Eriksson and Isaksson [221] demonstrated that proximal splitting could be used to solve pseudo-convex L_∞ problems in multi-view geometry, particularly discussing the problems of triangulation and structure from motion. They used the split Bregman algorithm to develop their method, however, this resulted in essentially the same iterates that ADMM would. Their primary goal was to demonstrate that proximal splitting methods produce efficient, scalable and straight-forward implementations of multi-view geometry problems, particularly structure from motion. In a continuation of this work, Eriksson et al. [222] developed a method for bundle adjustment with extremely high tolerance to outliers. They used proximal splitting methods to produce a least quantile of squares (LQS) estimator for an implementation of bundle adjustment with a theoretical breakdown point of 50% outliers. They demonstrated it working better than Tukey and Huber M-estimators for 15% outliers on two datasets. Finally, Eriksson et al. [223] demonstrated that these proximal splitting methods could result in a bundle adjustment algorithm that was suitable for parallel or distributed processing. By dividing up the structure from motion problem by disjoint partitions of images, their splitting method resulted in two sub-problems. The first was separable across the image partitions and the second was separable across the 3D world points. This allowed each of the sub-problems to be solved in a distributed manner. Their work on different facets of bundle adjustment with proximal methods demonstrated the multiple benefits of these methods. They firstly showed that proximal methods produced algorithms with excellent convergence properties, even for non-convex problems, they then showed that highly robust estimators could be developed in much more computationally feasible ways, and finally they demonstrated how proximal methods lead to robust distributed algorithms.

Distributed optimisation using proximal algorithms provides a clear framework for developing distributed algorithms across networks of nodes in a manner that has a well defined global objective function. As can be seen, it has been applied in a range of areas resulting in competitive robust estimators when compared to alternative distributed algorithms as well as robust and distributed computer vision algorithms. There has already been a number of applications of ADMM and other proximal methods to the problem of localisation through the bundle adjustment algorithm.

2.5.3 Distributed calibration and localisation

Distributed calibration and localisation of a camera sensor network is where the methods of multi-camera calibration that have been discussed are applied to a distributed CSN. In this case, there is no central node available to perform the full calculations and, therefore, the various distributed algorithms discussed in Section 2.5.2 are required to spread the calculation over each of the nodes. Some of the work in this area focuses on the full calibration and localisation problem by computing both intrinsic and extrinsic parameters, whereas other work focuses instead only on the localisation problem of the extrinsic parameters with the assumption that the camera is intrinsically calibrated ahead of time.

There has been much more work in the general task of distributed calibration and localisation. Early work used pairwise matched feature points and performed self-calibration locally at each node, which was then aligned into a global solution using average consensus. Devarajan and Radke [157, 158] presented a method using self-calibration based on the absolute quadric method of Triggs [47]. They connect each node to its neighbours in a cluster and perform a local self-calibration. Global agreement of the extrinsic calibration is achieved by aligning frames across the vision graph edges through a separate nonlinear least-squares problem that finds the transformation that best aligns two point clouds. They use unique identifiers to determine which node will act as the 'basis' for this transformation, in effect placing the lowest identified node at the origin of the coordinate system. They later expanded on their method by using belief propagation to improved the consistency of the local self-calibration estimations [156]. After their frame alignment process, they made sure that neighbours agreed on all parameters that they mutually estimated in the local stage through the message passing algorithm of belief propagation which determines the maximum likelihood estimation. Bringing the extrinsic calibration into global consensus is crucial for distributed computer vision as it prevents errors from accumulating and propagating throughout the system. Choudhary et al. also proposed a method for self-calibration of a distributed CSN using multi-layered belief propagation, specifically applying this to pan-tilt cameras. Their method divides the network into fully connected groups of three cameras where each camera in the clique estimates the parameters for all three cameras. They also discuss how the pan-tilt cameras can be actively controlled so that they belong to different cliques over the course of the calibration to facilitate the exchange of information over the entire network. This means that each camera node will have estimates for a number of other nodes that will not be perfectly consistent, which is where multi-layered belief propagation is used to bring these estimates into consensus across the entire network. They use a similar method to Devarajan and Radke for aligning coordinate frames, focusing on aligning cliques first.

An alternative form of distributed self-calibration was proposed by Elhamifar and Vidal [224] who considered the case of propagating calibration throughout a distributed CSN using the methods of self-calibration over a spanning tree of the network. They demonstrate that the nonlinear system of two camera self-calibration reduces to a straight forward linear system when one of the cameras has know intrinsic parameters. Therefore, along a spanning tree network, each uncalibrated node can easily determine its own intrinsic parameters and the extrinsic parameters relative to its calibrated neighbour. This propagates the calibration from a single calibrated camera out across the network towards the leaf nodes. By reducing the problem to a linear one, this opens up the possibility of using this type of algorithm on substantially lower-power hardware which is appealing for distributed robotic CSNs, however, the requirement for an already calibrated camera does limit its application. This is similar to the method of Vasconcelos et al. [79] discussed previously, where a new uncalibrated camera can be added to a network of calibrated cameras and have that calibration propagated to it. Their method relied on pairwise correspondences with at least two calibrated neighbours, whilst Elhamifar and Vidal's method only assumes that a single camera is calibrated. The method of Vasconcelos et al. can be equally applied to a distributed network as a centralised one as it only requires communication with those two direct neighbours.

Much of the work on calibration in distributed CSNs has focused on solely extrinsic calibration with the assumption that the intrinsic parameters are known. As mentioned in the discussion on self-calibration of a single camera, once the intrinsic parameters are known it is fairly simple to determine the extrinsic parameters. For the single-camera case these are the locations and orientations of each images from the moving camera, however, for the multi-camera case this is the localisation of each different camera. For self-calibration, once the intrinsic parameters are known the essential matrix between two views can be estimated in the same manner as the fundamental matrix [19], then this can be decomposed into the extrinsic parameters. For distributed CSNs, the estimations at each node might not be consistent or the sharing of such information might be too excessive to allow for these simple methods and, therefore, a variety of alternative distributed extrinsic calibration algorithms have been developed.

An early method for distributed localisation was presented by Mantzel et al. [225] for segments of networks that are sufficiently connected, which they refer to as micro-clusters. Their method requires that the network be comprised of fully-connected triangles that are connected to other triangles by at least two common nodes. Their method has one camera act as the origin of the coordinate frame and assumes that the distance to the second camera is one, estimating the extrinsic parameters of the second camera using the essential matrix decomposition method. From here, each camera estimates its own extrinsic parameters using point correspondences once it has two fully localised neighbours. Due to the connected nature of the micro-cluster, all nodes will eventually be localised. A drawback of this method is that it does not work for parts of the network that are not sufficiently connected.

Calibration objects have been utilised to improve the performance of some distributed methods. A method by Barton-Sweeney et al. [226] sees camera nodes with LEDs attached that are used to identify a given node in the field of view of another node. Therefore, each node can be localised using epipolar geometry, either from the fundamental matrix when two nodes both observe a common third node, or directly when two nodes also observe each other. Kurillo et al. [227] localised from pairwise point correspondences and epipolar geometry, with metric scale determined from a calibration object comprised of LEDs separated by a known length. Correspondences are also used to determine weighted adjacency on the vision graph, with Dijkstra's algorithm used to find the optimal paths on the vision graphs to transform local extrinsic estimations into a globally consistent localisation. Medeiros et al. [228] utilise pairwise correspondences of a moving object to perform relative extrinsic calibrations between neighbouring nodes, with known distances on the moving object being used to provide metric scale. To achieve consensus in a common reference frame, an information weighted average consensus is used with a given node transforming its local estimates into the coordinate frame of the neighbour with which it is currently communicating if that neighbour has a lower ID. At the end of this process, the nodes should all be in the same reference frame with agreement over common parameters. An interesting method by Pollok and Monari [229] localised a distributed CSN with non-overlapping FoVs by utilising the output of a SLAM algorithm run on a mobile camera passing through the same area. They rely on a keyframe-based SLAM algorithm to provide both a point cloud representation of the scene as well as information rich keyframes. Each fixed camera to be localised chooses keyframes that are most similar and determined its extrinsic parameters relative to the pose of the mobile robot at that frame. All of these methods utilise their calibration objects or prior information to simplify the problem and achieve a globally consistent localisation, with Kurillo et al. and Medeiros et al. deriving known metric scale from their constraints.

Tron and Vidal performed distributed localisation using matched feature points [8, 33]. They used average-consensus based gradient decent to perform optimisation on the manifold of $SE(3)$ poses. Ambiguities in the solution are solved by separating the problem into three steps — first rotations are made globally consistent, then translation and scene scale estimates, then finally all parameters are optimised. Their work is novel in showing how an algorithm as simple as average-consensus can be used to apply strict mathematical constraints on a complex optimisation problem.

As can be seen, the majority of these algorithms are based on self-calibration or epipolar geometry rather than any of the object-based calibration methods discussed prior. For the extrinsic-only methods where the intrinsic parameters are already known, this is the most straight-forward approach, even when metric scale is required, as it is fairly trivial to impose constraints of known lengths on such methods. This was seen in the calibration objects of Kurillo et al. and Medeiros et al. This, however, is not as simple when metric scale is require and the intrinsic parameters are not known. The methods discussed here that include estimation of intrinsic parameters either rely on self-calibration or the propagation of calibration from one or many already-calibration cameras.

2.5.4 Distributed multi-robot visual odometry and SLAM

Distributed multi-robot visual odometry and SLAM involves systems of multiple robots where each are running a local on-board VO or SLAM algorithm whose map is combined with or aligned to the other robots in the system through distributed process, such as the consensus methods discussed in Section 2.5.2, done so without the use of a centralised processor. There is substantial overlap between some of these methods and the distributed localisation (extrinsic calibration) discussed in Section 2.5.3, however, there are two main differences. Firstly, the localisation is continuous meaning that each node is moving and has multiple poses associated with it over time. Secondly, the localisation itself is just one aspect of the VO and SLAM pipeline that was discussed in Section 2.4.2, with map consistency, map alignment, loop closure, and relocalisation also being important features that need integrating into a distributed multi-robot system. In this section, some of the features of a centralised multi-robot SLAM system are discussed in how they related to map division that can be applied to distributed systems. Then various approaches to distributed map alignment and merging are discussed. Following this, methods for performing the localisation and bundle adjustment in a distributed manner that includes all connected robots are discussed and related to the distributed localisation seen in Section 2.5.3 and the distributed filters seen in Section 2.5.2. Finally, to tie these various approaches together, some notable full distributed SLAM systems are discussed.

Related aspects of centralised multi-robot SLAM

There are a range of centralised multi-robot approaches that have applicable features which can be useful for consideration in distributed systems. C2TAM [230] is an example of a centralised multi-robot SLAM system where a large portion of the processing is done on a cloud server. This method is built upon PTAM [20] which was the first SLAM system to separate the mapping and tracking processing into different threads. C2TAM takes this further and does the tracking thread locally on the robot but separates the mapping thread to a cloud server with the optimised map being downloaded to the robot after changes. The authors initially consider just the single-robot case where the expensive global bundle adjustment and map maintenance is done by the cloud server, however, for the multi-robot case, as well as the case where a single robot operates on different maps at different times, map fusion is done on the cloud and when an overlap is detected each robot downloads the new combined map. The authors argue that the centralised solution to map building and optimisation allows significantly larger amounts of data to be leveraged and for the full raw keyframes from all robots to be retained. This method demonstrates a similar approach as could be taken for distributed SLAM, with the tracking thread, or VO front-end, being done in the usual single-robot manner but the global mapping being separated into a distributed process. Similarly, Dong et al. [231] proposed a SLAM method that has qualities of centralised and distributed processing. The authors differentiate between full-SLAM, where map data is shared, and pose-SLAM, where only relative measurements between robots is established, with their focus being on multi-robot pose-SLAM. They demonstrate a 2D laser scanner SLAM system based on *expectation maximisation* (EM) where each robot solves for a subset of the graph based on their local pose-to-pose measurements and relevant inter-robot relative measurements. To obtain inter-robot correspondences, they use feature-base matching where scans from the robots are broadcast to all other robots in an assumed all-to-all wireless network, meaning that initially they are solving the centralised correspondence problem at all nodes. Despite this, they demonstrate the separability of the problem that can be solved in an entirely distributed manner that does not have all-to-all connections.

Another similar aspect of centralised systems that shows strong applicability to the distributed problem is the use of submapping algorithms for parallel out-of-core optimisation. Grisetti et al. [232] proposed a method for pose graph optimisation that uses multiple levels of 'coarseness', with the lowest level representing the original input and each level above that having nodes that represent subgraphs of the lower level. New measurements are added to the lowest level and either join an existing subgraph or start a new one. If it starts a new subgraph, the corresponding node is added at a higher level and so on up to the coarsest level. Optimisation is always done starting at the top of the hierarchy and the subgraphs corresponding to higher level nodes are only optimised on lower levels if their parent node is sufficiently changed. Although their method focuses on minimising the amount of fine-detail optimisation required, it also opens up the possibility of subgraph optimisation being performed at a separate processor for parallel processing. Similarly, Ni and Dellaert [233] proposed a method for partitioning the SLAM factor graph into a hierarchical structure using nested dissection where base nodes were optimised separately using a bottom-up approach. Furthermore, Zhao et al. [234] proposed a method for performing the inverse operation, submap joining, where they showed that it can be reduced to a linear least squares problem combined with frame transformations to produce a binary tree 'divide and conquer' method. This approach was highly efficient, however, the results differ from the global optimal solution. One can see how these submap approaches that allow for parallel optimisation of different parts of the graph could be applied to a distributed system where the global map is agreed upon through a distributed process, then partitioned so that each node can optimise a local portion, then distributed optimisation can be used to recombine the partitioned map.

Distributed map alignment and merging

An important aspect of distributed multi-robot SLAM that differentiates it from standard single-robot SLAM is the need to align and merge the various maps generated locally at each robot. Some systems utilise the merged maps to maintain a copy of the full global map locally at each robot node, whereas other systems only retain minimal information about the alignment and merging and instead focus on augmenting the local map with this information. The goal of having aligned and merged maps is that this allows the exploration of the robot system to be performed collaboratively.

Some of the early methods for map alignment and merging were done for collaborative SLAM systems that used rendezvous manoeuvres to assist the alignment process. Fox et al. [6] discussed cooperative exploration of multiple robots in a distributed setting where map merging occurred on a network of limited communication that provided each robot with access to a shared map for use in frontier exploration. Assuming that partial overlap exists between pairs of robots, their method had each robot explore until they are able to communicate with another robot at which point they exchange sensor data to estimate their relative positions, with rendezvous manoeuvre performed to verify this estimation hypothesis. Successful hypotheses are used to align and merge the maps that result in each robot becoming a part of a cluster that eventually spans the global map, and as robots join this cluster they are able to participate in the cooperative frontier exploration. Their method uses a particle filter approach for the tracking and relative estimations, and a graph-SLAM approach for map merging. Zhou and Roumeliotis [24] also designed a map alignment and merging system that includes rendezvous manoeuvres, with their method build around EKF-SLAM. In a similar manner to that of Fox et al., pairs of robots take bearing and distance measurements of each other to determine relative locations and, if successful, transform their state vector and covariances into a common frame and then combine their EKFs. Duplicate landmarks are combined using a nearest neighbour search. These two methods rely on a handful of assumptions. Firstly, they assume that the communication bandwidth can support transmitting their full maps; secondly, they assume that the system is free to perform rendezvous manoeuvres; and finally, they assume that each robot is capable of obtaining these relative measurements. With these very reasonable assumptions, the procedure to align and merge maps is straightforward. However, it is also interesting to consider how alignment and merging can be done without such assumptions.

Research has been done into methods that do not rely on those assumptions, which results in a less straightforward solution. Birk and Carpin [235] considered the case where the robots were unable to obtain relative measurements of each other and instead needed to register their maps in an alternative way. Their method used Gaussian random walks with an image similarity metric applied to 2D occupancy grids. This essentially treated the maps to be aligned and merged as images to be registered. The robots explore until their local maps are sufficiently large and then they attempt to register these occupancy grid maps with the other robots. Lazaro et al. [236] also considered map merging under fewer assumptions, focusing on communication bandwidth with a method for multi-robot map alignment and collaborative SLAM where ‘condensed measurements’ are transmitted instead of full local maps. When two robots are within communication range for the first time, they exchange their most recent laser scans and last N nodes of their graph. This allows each robot to align maps using a RANSAC procedure and then maintain an understanding of what node of the neighbouring robots it shares factor with. Using this information, all subsequent messages between robots are of the 'map maintenance' type and only contain a condensed version of the map with only the nodes with which the receiving robot shares factors. Aragues et al. [237] proposed a method based on average consensus that similarly relied less on communication ability and robot-to-robot measurements, however, their method also introduced a number of new assumptions. In their method, each robot builds and maintains its own local map independently and then periodically uses average consensus to construct a global map with the other robots. The local map is still retained and used for further local mapping and fusion to avoid 'double counting' of such information. The global map is found through average consensus using the assumption that all robots are operating with aligned reference frames and have no robot-to-robot measurements. By expressing the map in the *information filter* form, the rows and columns of the information matrix relating to poses can simply be broadcast between robots to form the global map and the remaining lower right portion relating to observed features can be computed using average consensus tracking. As mentioned, this method relies heavily on the robots having the same frame of reference at the beginning of operation.

These different approaches to map alignment and merging allow SLAM algorithms at each node to have their local graphs combined into a consistent global graph, or gain an understanding of how their local graph relates to a possible global graph. Through this, the local SLAM can be leveraged to perform cooperative exploration across the multiple robots and efficiently explore a given environment. This can be seen as a first step towards a fully distributed SLAM, in that the majority of the SLAM algorithm is the same as the single-robot case with the addition of a distributed alignment and merging step.

Distributed optimisation and bundle adjustment

Another aspect of distributed VO and SLAM is the global optimisation operations that occur, which can include pose graph optimisation and bundle adjustment. In the single-robot case, these are done over the poses of the robot over the course of its trajectory. In the multi-robot case, this is done over multiple trajectories as well as inter-robot measurements that relate the trajectories to each other. This is also closely related to the multi-view localisation problem discussed in Sections 2.3.3 and 2.5.3 with the main difference that each node has a trajectory of different poses associated with it, some of which have relationships to the poses of other trajectories.

The various distributed multi-view localisation problems that have been discussed are directly relevant to distributed pose graph optimisation. This problem has been addressed using the range of consensus methods that have been mentioned so far. Average consensus was utilised by Tron and Vidal [8] to develop a gradient decent algorithm for optimisation on Riemannian manifolds. Since the problem was separable across each camera node, it was able to be efficiently solved using average consensus and a three-stage process refining rotations first, then positions, then the full poses. Belief propagation was also used for multi-node localisation in the work of Ihler et al. [203] where edges between nodes represent distance estimates and the problem was solved effiently using nonparametric belief propagation. Furthermore, proximal methods have also been used to solve the full distributed bundle adjustment problem, as seen in the work of Eriksson et al. [223] where the demonstrated separation for parallel processing such that the subproblems that can be computed independently. This results in two steps to each iteration, the first is separable over disjoint partitions of images and the second is separable over the 3D points being tracked. These three different approaches to distributed pose graph optimisation and bundle adjustment could be directly applied to the case where multiple nodes correspond to the trajectory of a robot.

There has also been substantial research into developing methods explicitly for the multi-robot collaborative localisation problem. Nerurkar et al. [238] performed bundle adjustment using the conjugate gradient method which produced a distributed algorithm. Conjugate gradient solves the linear least squares update step of each non-linear optimisation iteration through an iterative process rather than inverting the Hessian. Their formulation allows each robot to compute only the parts of this inner iteration that relate to itself and then that is communicated to neighbours. Their method has only one robot pool the results for one of the steps and communicate the update back to the other robots, however, that is not strictly speaking distributed. On the other hand, most purely distributed approaches would compute this update at all nodes, which would be a minimal change to their algorithm. Knuth and Barooah [25] proposed a method for pose graph optimisation that was similar in construction to that of Tron and Vidal [8] by using optimisation on Riemannian manifolds, however, their solution is much closer to the centralised solution. They assume that each robot has an estimate of its current location as well as some number of inter-robot relative pose measurements. Robots communicate these two values to their neighbours where they are incorporated into a local pose graph optimisation that refines for just the local robot's pose. This method does not have the robots refine estimates of the poses of neighbours and therefore does not require any form of consensus. They then further extended their method to work with a range of inter-robot measurements beyond relative pose, including rotation, translation, distance, and bearing measurements [26]. Choudhary et al. [27] developed a method for pose graph optimisation that was more directly a distributed approach to the centralised objective function, with a focus on exchanging minimal information. They use a two stage approach that first estimates the rotations for a relaxed linear least squares problem and then estimates the full poses using a Gauss-Newton iteration, reducing the problem to two linear systems is a similar manner to how Tron and Vidal separated the problem into three stages. They demonstrate that this problem can be distributed across the robots, where each linear system is solved by the iterative Gauss-Seidel method separated by the block structure of the problem, resulting in an average consensus-like approach.

These methods demonstrate how the VO and SLAM pipeline can be adapted from the single-robot case to the multi-robot distributed case through the pose graph optimisation or bundle adjustment stages. Combined with map alignment and merging, this provides the necessary building blocks for distributed collaborative SLAM.

Full distributed SLAM systems

Features such as map alignment and merging, distributed bundle adjustment or pose graph optimisation, inter-robot loop closure, and relocalisation allow a single-robot visual odometry or SLAM system to become a fully-fledged distributed collaborative SLAM system. There have been a range of systems that deliver this complete SLAM feature-set, and their design choices highlight the necessary priorities of such systems. These priorities include communication constraints, global consistency, and managing information flow to avoid 'double counting' of measurements.

One form of distributed collaborative SLAM was DDF-SAM proposed by Cunningham et al. [28] whose method had three main modules: a local VO module that tracks the robot and produces a graphical map, a map alignment and merging module that communicates condensed maps to neighbours, and a distributed pose graph optimisation module that operates on neighbourhoods. Their neighbourhood maps are highly condensed factor graphs that only involve data relating to landmarks, greatly reducing the data required to be sent between robots. These landmark-only condensed maps are shared between robots so that each robot can combine them for neighbourhood optimisation. Since each condensed graph was linearised in different reference frames, the combined graph maintains their separate frames and relates them by 'neighbourhood frame' landmark nodes which introduce a number of equality constraints to the overall optimisation problem. The separation between the full local graph and neighbourhood graph is important to ensure that 'double counting' does not occur. This is where measurements from one node are sent to another and incorporated into the receiving node's estimates which ultimately affects measurements communicated back to the original node, over-weighting the influence of those measurements. This, however, means that robots need to maintain two graphs and that the optimisation over the neighbourhood cannot be utilised by the local map. They later address these problems in DDF-SAM 2.0 [9], where the condensed neighbourhood map is used to augment the full local map with the introduction of subtractive 'anti-factors' introduced to counter the effect of double counting.

Paull et al. [239] proposed a similar distributed SLAM system that was designed specifically for autonomous underwater vehicles which operate in an extremely communication-constrained environment. Condensed graphs are generated in a similar manner to Cunningham et al. [28], however, the poses to be marginalised out is determined by the last successful communication event. Therefore, the subsequent pose information is retained and sent. Furthermore, graph sparsification techniques are utilised to minimise the amount of data required to communicate compared to the normally dense marginalisation matrix. All measurements are designed to be relative to the last successful transmission so that packet loss does not impact consistency of the graph. Like with DDF-SAM, double-counting is avoided by only transmitting locally acquired data.

There has also been recent research on distributed visual SLAM methods that are specifically designed around indirect inter-robot estimations from visual features. Cieslewski et al. [240] presented a method that finds these indirect measurements in a similar manner to loop closure in single-robot visual SLAM. Inter-robot correspondences are first established using place recognition on compact whole-image descriptors, then if a correspondence is established the relative pose can be computed from the visual data association. To minimise communication for place recognition, each robot will only send its descriptor to a single other robot based on pre-assigned descriptor ranges, however, to propagate this throughout the network the receiving robot keeps this descriptor and can use it as a reply to another robot in a later place recognition operation. A downside of their method, however, is that all-to-all connections between the robots is assumed. Their approach to distributed optimisation is to adopt the approach of Choudhary et al. [27] and use a distributed Gauss-Seidel algorithm applied to a two-stage linear relaxation of the pose graph optimisation problem. In a similar manner, Lajoie et al. [241] proposed DOOR-SLAM which has many of the same features, however, has a much stronger focus on outlier rejection for inter-robot loop closures. For place recognition, the same whole-image descriptor method of Cieslewski et al. is used, however, rather than communicating this to a single robot based on pre-assigned descriptor ranges all descriptors accrued between communication events is sent to the target robot when a rendezvous event occurs. The relative pose estimation is then made based on visual features and verified using RANSAC with an inlier threshold. Their distributed pose graph optimisation is also based on the two-stage distributed Gauss-Seidel approach of Choudhary et al. Additionally, during the pose graph optimisation, spurious inter-robot loop closures are rejected if they are identified as outliers when checked for pairwise consistency.

These methods demonstrate how distributed collaborative SLAM can be constructed from the framework of a standard single-robot visual odometry or SLAM system with the addition of a number of distributed data sharing and consensus-based features. One such feature is a map alignment, merging, or sharing method that allows local map data to be integrated into the estimations of other nodes, such as in the condensed factor graph maps of Cunningham et al. [28] or in the visual feature loop closure systems of Cieslewski et al. [240]. The other main consensus-based feature is distributed optimisation such as the approach of Choudhary et al. [27]. The prevalent theme of how these systems are designed is managing communication constraints whilst still allowing the effective sharing of data throughout the system.

2.5.5 Summary and open problems

Through the exploration of distributed algorithm and their application to calibration and SLAM, this section has identified some key insights into current distributed calibration and SLAM systems. The three main categories of distributed algorithms considered were average consensus, belief propagation, and proximal splitting methods for distributed optimisation. Average consensus was most useful when the communication component was reduced to a simple average, however, work such as was seen in that of Tron and Vidal [8] demonstrated how robust global objectives could be achieved through average consensus methods. Belief propagation is designed around inference on graphical representations of the problem, but it has been shown that it can be interpreted as an average consensus problem. The results achieved through belief propagation are only approximate on loopy graphs, however, belief propagation has fast convergence properties when it is able to converge and it has been shown that the faster it converges the more accurate these approximations are [201]. Proximal splitting methods provided a class of powerful and highly general meta-algorithms for deriving distributed algorithms with clear non-linear global objectives and excellent convergence properties for problems where the objective was separable across the graph. These methods provided a more straightforward approach to distributed optimisation problems than average consensus or belief propagation, however, they do ultimately rely on distributed averaging and therefore would include one of the two alternatives as a component of the overall algorithm.

Regarding calibration, the majority of current distributed calibration and localisation algorithms are built around self-calibration only. Furthermore, most of the algorithms focused only on the localisation component of the problem. The work of Devarajan and Radke [157] is an example of a complete distributed calibration and localisation system, and it would be interesting to see similar approaches taken for methods that utilise a calibration object such that metric calibration to a known scale is possible. A drawback to their method is that they ultimately take the distributed average of the local objectives through Gaussian belief propagation, which is not equivalent to a global bundle adjustment objective. Therefore, it would be beneficial to see an alternative algorithm that utilises the average consensus approach of Tron and Vidal [8] for the full calibration problem or, perhaps more straightforward, the application of proximal splitting methods to ensure a correct global objective.

Regarding distributed SLAM methods, the areas of focus for distributed processing include map alignment and merging, pose graph optimisation and global bundle adjustment. Map alignment and merging is generally done by estimating relative poses from the partial overlap of the maps, which can be created through rendezvous manoeuvres, then merging corresponding points through nearest-neighbour or feature matching methods, as was seen in the work of Fox et al. [6] and Zhou and Roumeliotis [24]. Distributed pose graph optimisation and bundle adjustment is generally done in two ways: firstly, by exchanging information with direct neighbours to allow a reduced form of the problem to be constructed locally on the neighbourhood only, as was seen in the work of Knuth and Barooah [25], or by leveraging the separability of the optimisation problem to reduce it to iterations of local optimisation problems and distributed averages, as seen in the work of Tron and Vidal [8] and Choudhary et al. [27]. Applying these distributed methods to the full SLAM system, the common approach is to build upon the standard single-robot SLAM framework, similar to that discussed in Section 2.4.2, with the addition of map alignment, merging, pose graph optimisation, and bundle adjustment approaches, as seen in DDF-SAM [9] where any local VO system that produces a factor graph map can be utilised. There is still room for the development of holistic distributed SLAM systems that utilise homogeneous data-types and methodologies to more tightly integrate the underlying single-robot VO with the overall distributed SLAM, such as utilising the underlying loop closure and multi-map methodologies of ORB-SLAM [100] for inter-robot alignment and map merging with the global bundle adjustment objective being distributed in an approach like that of Tron and Vidal [8] or Choudhary et al. [27].

From this section, the following open problems related to the pipeline of this thesis can be identified:

- Feature-based vision graph estimation could be done by combining the reduced communication load of feature digests and the limited communication events of a communication graph-based.

- Vision graph formation based on calibration objects or as a continuous process produced by the coherent motion of SLAM is worth exploring.
- Existing distributed calibration and localisation methods are generally based on self-calibration. Exploration of similar 1D approaches could yield more accurate methods that provide scene scale.
- Distributed optimisations for SLAM currently involve trade-offs between communication constraints and effective sharing of crucial data.
- Distributed SLAM systems do not take direct advantage of the underlying SLAM algorithms, but are build on top of them in an interchangeable way.

2.6 Chapter summary and proposed framework

This chapter has analysed the breadth of research in the areas of calibration, localisation, visual odometry, and SLAM, as well as their applications to distributed robotic networks. Section 2.3 explored the tightly coupled problems of calibration and localisation, also referred to as intrinsic and extrinsic calibration, both in the context of individual cameras and multi-camera networks. This explored the rich history of the field and the various calibration objects, or lack of calibration objects, that have been the basis of numerous methods. Section 2.4 explored the development of visual odometry and SLAM systems over the years, what the difference are between the two classes of algorithms, as well as what technologies and paradigms are commonly utilised. This section developed a generalised framework for keyframe-based VO and SLAM, exploring how different algorithms implement the key components required, as well as what components are optionally included. Section 2.5 brought these areas of consideration together in their application to distributed networks. Firstly, the structure of a distributed camera sensor network was defined, then a number of key algorithms that form the basis of distributed processing were presented, and finally specific work on distributed calibration, localisation, visual odometry, and SLAM were explored both in relation to the base algorithms that were presented in Sections 2.3 and 2.4, as well as with regard to the underlying distributed processing methodologies that were presented. In this section, the key findings from the previous sections are summarised with a focus on the overall trend and avenues for future work, then they are related to a general framework for a distributed robotic SLAM system that informs the work presented in subsequent chapters.

Section 2.3 explored the different types of calibration and their application to multi-camera systems. Self-calibration is popular due to the lack of a calibration object; however, it is unable to determine the scale of a scene without additional information. With regard to calibration objects, 3D objects are designed to a high precision, but this introduces complexity and is generally unsuitable for multi-camera systems; 2D objects are much more flexible but still have some self-occlusion issues; and 1D objects remove these issues in multi-camera systems but have not yet seen as wide adoptions, mainly due to the sheer simplicity of the 2D calibration object. For these multi-camera systems, 1D and self-calibration were identified as the most suitable. However, open issues were identified relating to the formulation of the 1D calibration problem, simplification for the end-user, and the potential of novel calibration object types.

Section 2.4 discussed the two main paradigm splits in the areas of VO and SLAM, direct versus indirect, and dense versus sparse, also discussing the range of algorithms that don't fall neatly into these distinctions. Indirect methods are more robust for matching between wide baselines and different mapping sessions; however, direct methods allow for more computationally efficient methods. Sparse methods similarly offer greater computational efficiency, while dense methods utilise a greater portion of the image frames. The other main component of Section 2.4 was the application of a generalised keyframe SLAM framework to the many algorithms discussed, identifying the two local threads that were important to VO algorithms and the additional one or two threads used in full SLAM systems. The analysis of this section identified open problems and opportunities for future research relating to bridging the gap between direct and indirect methods: introducing direct components to the loop closure and relocalisation procedures, utilising feature descriptors in direct pipelines, and examining the use of direct matching for wide baselines.

Section 2.5 explored the fundamental distributed algorithms that have been used to create distributed implementations of calibration, localisation, VO, and SLAM. Average consensus is a method that is most useful for linear systems, however, complicated global objectives have been successfully implemented using this approach. Belief propagation has its strengths in inference problems but has significant overlap with average consensus. Distributed optimisation, on the other hand, reveals opportunities for highly complex nonlinear objectives to be distributed across many nodes. Section 2.5 also explored the application of these algorithms to calibration, localisation, VO, and SLAM. For calibration, self-calibration has received most of the focus with implementations based on both average consensus and belief propagation. Regarding SLAM, implementations generally apply the relevant distributed algorithms to components of map alignment and merging, pose graph optimisation, and global bundle adjustment.

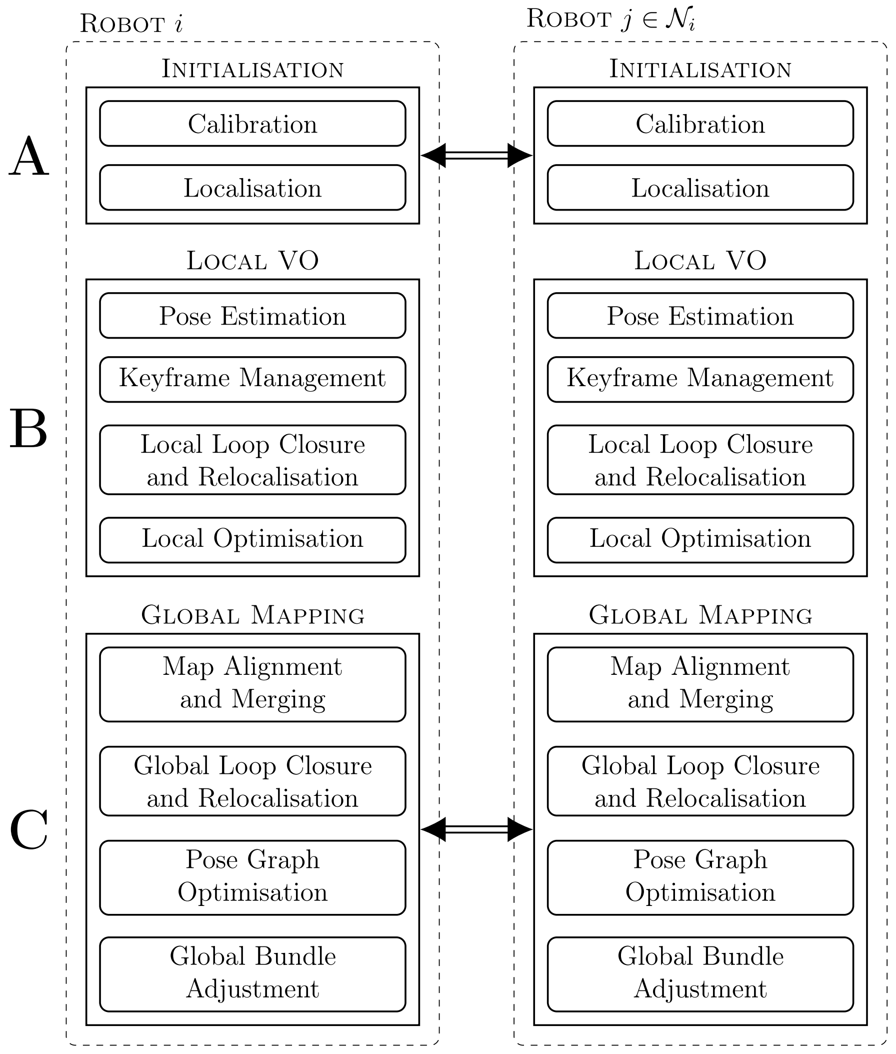


Figure 2.28: A framework for a distributed SLAM system built around an underlying single-robot VO algorithm. Calibration and localisation is done in a distributed manner to provide a known static initialisation, then local VO is performed without communication, and finally the local estimates are brought to global alignment through distributed methods.

2.6.1 Applications to the present work

The research and trends discussed in this chapter have shown that a general distributed full SLAM system with included calibration and static localisation can be viewed with the framework given in Figure 2.28, which is an extension of the framework introduced in Figure 1.2 in Section 1.2. As can be seen, communication between neighbouring robots is done during the initialisation for the purposes of calibration and consistent static localisation. Following this, each robot performs local VO corresponding to the methods discussed in Section 2.4, which is done in the standard single-robot manner without communication to neighbours. Then, for the global mapping stage, neighbour-based communication and distributed algorithms are utilised to bring the system into global consensus.

This leads to the identification of three main problems that are the focus of subsequent chapters. Firstly, relating to problem (A) in Figure 2.28, the problem of distributed calibration and localisation as a suitable initialisation is considered in Chapters 3 and 4. Most distributed calibration and localisation algorithms are based on self-calibration, and often only focus on the extrinsic calibration. In Chapter 3, the adaption of a method similar to Devarajan and Radke [157] is considered for use with 1D calibration, as this provides metric calibration to a known scale using a form of calibration that does not have issues with self-occlusion of the calibration object. Chapter 4 then improves upon this system by utilising the general motion multi-view 1D calibration of de França et al. [87], however, in a more simplified form that condenses the nonlinear optimisation to a single stage, then improving the distributed consensus algorithm by utilising ADMM with Gaussian belief propagation to split the global objective function into local objectives combined through distributed averaging. Next, relating to problem (B), a single-robot visual odometry method is considered in Chapter 5 with the goals of combining the computation efficiency of direct methods with the wide-baseline matching possibilities of indirect methods. This is designed to produce a method that can run on the lower-powered hardware that would be found on-board on robotic vision platforms whilst ensuring that the local and distributed data-types remain homogeneous. Finally, considering problem (C), Chapter 6 explores a method of accelerating the convergence of Gaussian belief propagation through edge-weights such that distributed pose graph optimisation utilising that class of distributed algorithms can have higher accuracy results.

Chapter 3

Distributed one-dimensional camera calibration and localisation with Gaussian belief propagation[a]

Contents

3.1 Summary of contributions

- One-dimensional camera calibration is used to provide an initial intrinsic calibration at each camera node in a distributed CSN.
- Observations of the calibration object points are compared and used to initialise a undirected vision graph to a regular lattice structure.

[a]Chapter adapted from "B. Halloran *et al.*, 'Distributed one dimensional calibration and localisation of a camera sensor network,' in *Proceedings of ICIC 2017: Intelligent Computing Theories and Application*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2017, pp. 581–593, ISBN: 978-3-319-63312-1. DOI: `10.1007/978-3-319-63312-1_51`"

- Comparisons of 3D world points between close neighbours are used to determine an initial estimate for relative poses.
- Local neighbourhood-based bundle adjustment is performed and brought into global consensus using Gaussian belief propagation.
- Evaluations are done on simulated and real data, demonstrating that the distributed algorithm performs similarly to a centralised counterpart across a range of noise levels.

3.2 Introduction

This chapter explores a suitable initialisation stage for the three-stage distributed robotic vision pipeline presented in Figures 1.2 and 2.28, which has the overall goals of accurate calibration, localisation, and mapping.

Calibration and localisation of a CSN is an important first step for many higher-level computer vision tasks, allowing 3D information to be derived from 2D images. Extensive work has gone into studying calibration and localisation, however, most multi-view solutions require a centralised processor with access to data from all camera nodes, whilst existing distributed algorithms are generally slower and don't recover the scale of the scene due to their reliance on self-calibration. Distributed algorithms are becoming increasingly important in CSNs where centralised processing is not resistant to node failure and communication with a central node can be expensive, such as in battery powered applications. This promotes the need for a simpler distributed algorithm that can accurately calibrate and localise a CSN to known scale.

The most common calibration method is '2D' calibration, where each camera observes a planar checkerboard pattern at several arbitrary orientations [59]. This method is popular due to its simplicity, however, it's not suitable for large CSNs due to the planar pattern's self-occlusion at wider angles. Another popular method is 'self-calibration', which does not use a calibration object, but rather relies on constraints in the scene [17]. Here, each camera matches feature points from a static scene and solves the *structure from motion* (SfM) problem to calibrate and localise each node. This method is more suited to large CSNs than 2D calibration as there is no self-occlusion, however, it is more computationally complex and only calibrates and localises the system up to an unknown scale, which is insufficient for many applications.

The calibration method being utilised in this chapter is known as '1D' calibration, which involves observing three or more collinear 3D points where two of the points are moving and one remains fixed [15]. This method also doesn't have self-occlusion problems compared to 2D calibration and determines the scale of the scene compared to self-calibration. The other major component of our algorithm is *Gaussian belief propagation* (GaBP) [197, 242]. Here, each node on a probabilistic graph sends its neighbours messages based on initial potentials and current belief, which iteratively form new beliefs until the convergence. GaBP models the variables as Gaussian distributions, such that each message is simply two scalars representing the mean and variance of the density [200].

These features are combined to provide a distributed calibration and localisation algorithm that can accurately initialise a robotic vision platform, being suitable for use in the first stage of the pipeline being discussed in this dissertation.

3.2.1 Related Work

First proposed by Zhang, 1D calibration has received many improvements to greatly improve its accuracy [15]. Hammarstendt et al. simplified the closed-form solution of the problem and analysed critical configurations to identify the classes of motion that successfully solved the calibration [68]. Other researchers have relaxed the fixed-point constraint to allow for planar motion or motion under gravity for the calibration object [69–71]. De França et al. improved calibration accuracy by normalising the image points to improve numerical conditioning [72] and Shi et al. improved accuracy with a weighted equation based on orientation of the 1D object and the relative projective depths [73]. Recently Wang et al. demonstrated that a linear matrix inequality (LMI) relaxation algorithm could replace the non-linear optimisation, speeding up the algorithm [243]. There have also been improvements that allow for general motion of the 1D object in multi-view systems, however, the results are not yet as accurate as fixed-point methods [85–87].

There has also been work on distributed calibration algorithms, generally based on self-calibration. Devarajan et al. proposed a method that operates on a vision graph, which models the system as a Markov random field (MRF) where edges represent overlap in the fields of view [157, 158]. In their method, each node builds a cluster with its neighbours and performs SfM on these clusters. They later brought these estimates to global consensus through belief propagation [7, 156]. Tron and Vidal proposed an algorithm whereby pairwise pose estimates of a CSN were brought to alignment using an average consensus-based gradient descent algorithm which operated on the Riemannian manifold of the poses [8, 33].

3.2.2 Contributions

The proposed algorithm in this chapter consists of modelling the network as an MRF. Firstly, robust local 1D calibration is performed at each node to estimate the intrinsic parameters of the local node as well as the 3D points of the calibration object. The structure of the calibration object and the estimated 3D points are then utilised to construct a regular lattice-structured vision graph based on nearest-neighbours rotation-wise and a cluster-based bundle adjustment is performed such that each node obtains estimates of their cluster's extrinsic parameters. Finally, Gaussian belief propagation is used to align the estimates into global consensus. This produced two versions of the algorithm: a centralised version where each node performs the local calibration then a global bundle adjustment is performed, and a distributed algorithm where each node instead performs the cluster-based bundle adjustment with direct neighbours and the distributed consensus algorithm is used for alignment. Experimental tests were performed to compare these two alternatives, verifying that the distributed algorithm does not result in significant losses in accuracy.

The remainder of the chapter is organised as follows: Section 3.3 reviews the preliminaries for the graph and camera models; Section 3.4 describes the local 1D calibration algorithm; Section 3.5 explores the vision graph construction and the initialisation of pose estimates; Section 3.6 describes the application of Gaussian belief propagation; Section 3.7 presents the experimental results and analyses the performance; and finally, Section 3.8 summarises and concludes the chapter.

3.3 Preliminaries

In Section 2.3.1 the camera calibration problem was introduced and in Section 2.5.1 the representation of a CSN as two undirected graphs was introduced. However, in the following section, the important aspects of these preliminary concepts are reintroduced for clarity.

3.3.1 Camera sensor networks as Markov random fields

Section 2.5.1 introduced the *communication* graph, $\mathcal{G}_C = \{\mathcal{V}, \mathcal{E}_C\}$, and the *vision* graph, $\mathcal{G}_V = \{\mathcal{V}, \mathcal{E}_V\}$. In this chapter, the algorithms proposed operate entirely on the vision graph, however, knowledge of the communication graph would be required for implementation on a physical system. This chapter assumes that the ability is available to route information across the communication graph to the required neighbour on the vision graph. For the vision graph, $\mathcal{E}_V \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges containing $(i, j) \in \mathcal{E}_V$ pairs of camera nodes with sufficiently overlapping fields of view. We denote the direct neighbours of node i as $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}_V\}$ and the degree of a node is given by $d_i = |\mathcal{N}_i|$. Figure 2.24 in Section 2.5.1 demonstrates the relationship between the fields of view and the vision graph.

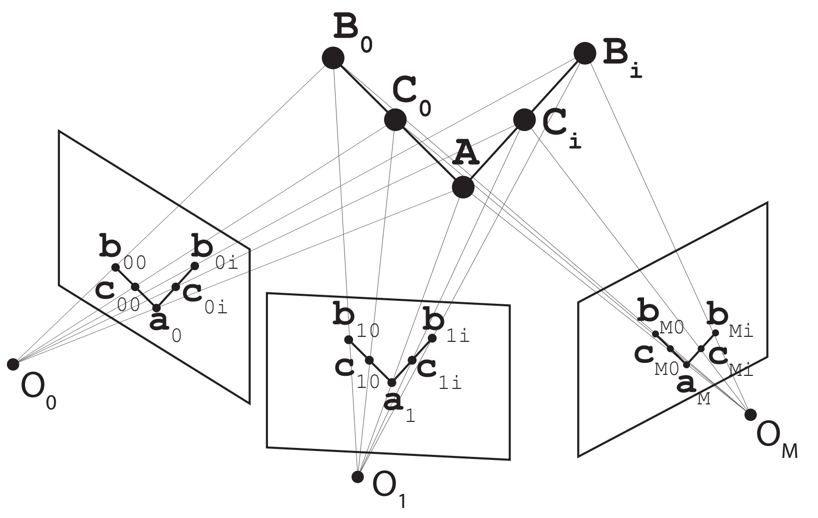


Figure 3.1: The one-dimensional calibration object shown at two positions, and its projections into three cameras.

3.3.2 The pinhole camera model

Considering the 2D point on the image plane $\mathbf{m} = [x, y]^T$ as a projection of the 3D world point $\mathbf{M} = [X, Y, Z]^T$, we represent these in homogeneous form as $\tilde{\mathbf{m}} = [x, y, 1]^T$ and $\tilde{\mathbf{M}} = [X, Y, Z, 1]^T$ respectively. The projection of the 3D world point into the image plane to produce the 2D image point is given by

$$\tilde{\mathbf{m}} \simeq \mathbf{K}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{M}}, \qquad \text{where} \quad \mathbf{K} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

The extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation of each camera in the world coordinate system. In the intrinsic matrix $\mathbf{K}$, α and β are scale factors along the x- and y-axes of the image, (u_0, v_0) is the principal point, and γ is the skew which is normally assumed to be zero.

3.4 One-dimensional camera calibration

Firstly, local calibration is performed at each node to determine the intrinsic parameters and 3D world points. A one-dimensional calibration object consisting of three collinear points is used, where one of the points is fixed. The input to the algorithm is the image points taken of the calibration object at N different orientations, rotated arbitrarily about its fixed point. These images are taken from M camera nodes, as shown in Figure 3.1.

3.4.1 Zhang's One-dimensional calibration algorithm

The calibration algorithm used in this chapter is based on that proposed by Zhang [15] with two improvements. Firstly, the fundamentals of the algrithms are presented. Consider the three collinear 3D world points $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, where $\mathbf{A}$ is a fixed point and $\mathbf{C}$ lies between $\mathbf{A}$ and $\mathbf{B}$ at a known distance. The known length between $\mathbf{A}$ and $\mathbf{B}$ is given by

$$L = ||\mathbf{B} - \mathbf{A}||_2. \tag{3.2}$$

The point $\mathbf{C}$ is found using the known ratios of λ_A and λ_B,

$$\mathbf{C} = \lambda_A \mathbf{A} + \lambda_B \mathbf{B}. \tag{3.3}$$

The 3D points are defined to be in the camera coordinate system such that $[\mathbf{R}|\mathbf{t}] = [\mathbf{I}|\mathbf{0}]$. Therefore, the corresponding image points $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$, and $\tilde{\mathbf{c}}$ are simply related to their respective 3D world points by the intrinsic matrix and some unknown projective depths:

$$\mathbf{A} = z_A \mathbf{K}^{-1}\tilde{\mathbf{a}} \tag{3.4}$$

$$\mathbf{B} = z_B \mathbf{K}^{-1}\tilde{\mathbf{b}} \tag{3.5}$$

$$\mathbf{C} = z_C \mathbf{K}^{-1}\tilde{\mathbf{c}}. \tag{3.6}$$

Substituting these expressions into Equation 3.3 and rearranging allows us to relate the projective depths of the points at either end of the calibration object using

$$z_B = -z_A \cdot \frac{\lambda_A(\tilde{\mathbf{a}} \times \tilde{\mathbf{c}}) \cdot (\tilde{\mathbf{b}} \times \tilde{\mathbf{c}})}{\lambda_B(\tilde{\mathbf{b}} \times \tilde{\mathbf{c}}) \cdot (\tilde{\mathbf{b}} \times \tilde{\mathbf{c}})}. \tag{3.7}$$

Then, by substituting this relationship and the world point projections of Equations 3.4 to 3.6 into Equation 3.2, the expression for the length of the calibration object is given in Equation 3.8

$$z_A^2 \mathbf{h}^T \mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h} = L^2 \tag{3.8}$$

$$\text{where} \quad \mathbf{h} = \tilde{\mathbf{a}} + \frac{\lambda_A(\tilde{\mathbf{a}} \times \tilde{\mathbf{c}}) \cdot (\tilde{\mathbf{b}} \times \tilde{\mathbf{c}})}{\lambda_B(\tilde{\mathbf{b}} \times \tilde{\mathbf{c}}) \cdot (\tilde{\mathbf{b}} \times \tilde{\mathbf{c}})}\tilde{\mathbf{b}}.$$

Now, consider the vector $\mathbf{\Omega} = [\omega_0\ \omega_1\ \omega_2\ \omega_3\ \omega_4\ \omega_5]^T$ which is taken from the unique elements of the symmetrical *image of the absolute conic*,

$$\omega = \mathbf{K}^{-T}\mathbf{K}^{-1} = \begin{bmatrix} \omega_0 & \omega_1 & \omega_3 \\ \omega_1 & \omega_2 & \omega_4 \\ \omega_3 & \omega_4 & \omega_5 \end{bmatrix}.$$

If we define $\mathbf{h}_i = [h_{i,0}\ h_{i,1}\ h_{i,2}]$ for each image i and $\mathbf{x} = z_A\boldsymbol{\Omega}$, then we can write Equation 3.8 for each image of the calibration object as

$$\mathbf{v}_i\mathbf{x} = L^2\mathbf{1} \tag{3.9}$$

$$\text{where} \quad \mathbf{v}_i = \begin{bmatrix} h_{i,0}^2 & 2h_{i,0}h_{i,1} & h_{i,1}^2 & 2h_{i,0}h_{i,2} & 2h_{i,1}h_{i,2} & h_{i,2}^2 \end{bmatrix}^T.$$

In Equation 3.9, an appropriately sized vector of ones is given by $\mathbf{1}$. For N images, each instance of Equation 3.9 is stacked to get the $N\times 6$ matrix $\mathbf{V} = [\mathbf{v}_0\, \mathbf{v}_1\, \ldots\, \mathbf{v}_{N-1}]$, which produces the linear least squares problem for $\mathbf{x}$ given by

$$\mathbf{V}\mathbf{x} = L^2\mathbf{1}. \tag{3.10}$$

This equation is solved for $\mathbf{x}$ using linear least squares, then the intrinsic parameters and projective depth of the fixed point, z_A, can be determined by Cholesky decomposition of $\mathbf{x}$. Following this, the projective depth of the moving end, z_B, can be determined for each image based on Equation 3.7 and therefore the full set of 3D world points can be determined using Equations 3.4 to 3.6.

Finally, these estimated parameters can be refined using nonlinear optimisation. However, the objective that was given in Equation 2.11 can be simplified using the constraints of the calibration object. The optimisation in Equation 2.11, discussed in Section 2.3.2, refines $8+6N$ parameters with five parameters for the intrinsic matrix, three for the location of the fixed point, and six further points for the location of points $\mathbf{B}_i$ and $\mathbf{C}_i$ for each image i. Instead, the number of parameters can be reduced to $8+2N$ by encoding the two moving points by the spherical coordinates θ_i and ϕ_i according to the relationship

$$\mathbf{B}_i = \mathbf{A} + L\begin{bmatrix} \sin\theta_i\cos\phi_i \\ \sin\theta_i\sin\phi_i \\ \cos\theta_i \end{bmatrix}. \tag{3.11}$$

This results in the nonlinear minimisation problem given in Equation 3.12, where π_A, π_B, and π_C are the functions projecting points $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ respectively into the image plane according to $\mathbf{K}$, using spherical coordinates for $\mathbf{B}$ and $\mathbf{C}$.

$$\begin{aligned}\{\mathbf{K}^*, \mathbf{A}^*, \theta_i^*, \phi_i^*\} = \underset{\{\mathbf{K},\mathbf{A},\theta_i,\phi_i\}}{\arg\min} \sum_{i=0}^{N-1} \frac{1}{2}(||\mathbf{a}_i - \pi_A(\mathbf{A};\mathbf{K})||_2^2 \\ + ||\mathbf{b}_i - \pi_B(\theta_i,\phi_i;\mathbf{K})||_2^2 + ||\mathbf{c}_i - \pi_C(\theta_i,\phi_i;\mathbf{K})||_2^2)\end{aligned} \tag{3.12}$$

3.4.2 Improvements to Zhang's Method

The algorithm presented in this chapter utilises two major improvements based on the work of de França et al. [72] as well as Shi et al. [73]. To improve the numerical conditioning of the linear least squares calibration problem, image data can be normalised using a transformation matrix $\mathbf{S}$ based on the image size to obtain the normalised homogeneous image points $\hat{\mathbf{m}}$, with the inverse transformation being applied to the resulting parameters $\hat{\mathbf{K}}$ to recover the non-normalised intrinsic parameters.

$$\begin{aligned} \hat{\mathbf{m}} &= \mathbf{S}\tilde{\mathbf{m}} \\ \mathbf{K} &= \mathbf{S}^{-1}\hat{\mathbf{K}} \\ \text{where} \quad \mathbf{S} &= \begin{bmatrix} 2/width & 0 & -1 \\ 0 & 2/height & -1 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Then, to further improve the quality of the linear least squares estimation, a weighting system can be used based on the relative projective depths from Equation 3.7 as well as the distance between image points $\mathbf{a}_i$ and $\mathbf{b}_i$.

$$\begin{aligned} \beta_i &= \frac{\lambda_A(\tilde{\mathbf{a}_i} \times \tilde{\mathbf{c}_i}) \cdot (\tilde{\mathbf{b}_i} \times \tilde{\mathbf{c}_i})}{\lambda_B(\tilde{\mathbf{b}_i} \times \tilde{\mathbf{c}_i}) \cdot (\tilde{\mathbf{b}_i} \times \tilde{\mathbf{c}_i})} \\ w_i &= \frac{||\mathbf{a}_i - \mathbf{b}_i||_2}{\beta_i^2} \end{aligned}$$

For the N images, the weighting matrix is defined as the diagonal matrix $\mathbf{W} = diag(w_1, w_2, \ldots, w_N)$. Using this, the linear least-squares problem in Equation 3.10 is replaced with the weighted and normalisation version in Equation 3.13.

$$\mathbf{WVx} = \mathbf{W}(L^2\mathbf{1}) \tag{3.13}$$

3.5 Initialising the distributed network

At this point in the algorithm, each node has independently determined their own intrinsic parameters based on their observations of the same calibration object points as the other camera nodes. The estimations of these 3D world points can now be used to initialise the vision graph and estimate the extrinsic parameters between the camera nodes.

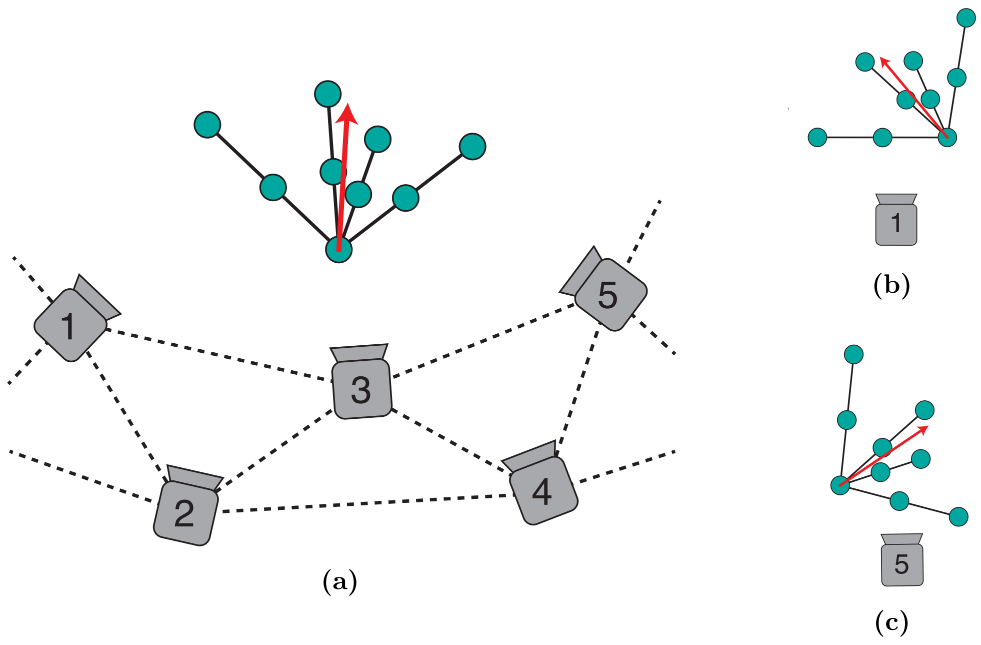


Figure 3.2: Example distributed network of at least five cameras, shown connected in a lattice with the four nearest neighbours in (a). Each camera estimates the principle spherical coordinates by averaging each calibration object observation. This is shown from the point-of-view of (b) Camera 1, and (c) Camera 5.

3.5.1 Connecting the vision graph

Now that the intrinsic parameters and 3D points are estimated, the vision graph of the system is initialised to determine the neighbours of each node. The vision graph is an undirected graph with cameras as nodes and edges representing an overlap in fields of view. Since all cameras are viewing the same object, we could assume a fully connected vision graph, however an overly connected graph can prevent the belief propagation from converging. We limited these connections to the four nearest neighbours based on spherical coordinates. Each node j calculates its 'principal spherical coordinates' from the spherical coordinates of each calibration object orientation i by averaging all $\theta_i j$ and $\phi_i j$, and then the closest four neighbours in rotation-wise sense are determined from the difference between principal spherical coordinates.

$$\begin{bmatrix} \Theta_j \\ \Phi_j \end{bmatrix} = \frac{1}{N} \sum_{i=0}^{N-1} \begin{bmatrix} \theta_{ij} \\ \phi_{ij} \end{bmatrix} \tag{3.14}$$

If the cameras are arranged around the calibration object with roughly even spacing, this results in a regular lattice graph with each node connected to the two nodes either side, as shown in the example of Figure 3.2.

3.5.2 Estimating extrinsic parameters

The estimated calibration object points are also used to determined relative poses between camera nodes. A node considers its neighbours on the vision graph as a cluster centred about itself and calculates the poses of the cluster, relative to itself, by comparing the calibration object world points. Firstly, it calculates the *centre of gravity*, $\mathbf{G}_j$ of the world points for each node in its cluster,

$$\mathbf{G}_j = \frac{1}{3N}\sum_{i=0}^{N-1}(\mathbf{A}_j + \mathbf{B}_{ij} + \mathbf{C}_{ij}).$$

To find the rotation and translation between node j and a neighbouring node $k \in \mathcal{N}_j$, we find the covariance, $\mathbf{\Sigma}$ between the points of the two nodes and apply singular value decomposition.

$$\begin{aligned}
\mathbf{\Sigma}_{jk} &= \sum_{i=0}^{3N-1}(\mathbf{M}_{ij} - \mathbf{G}_j)(\mathbf{M}_{ik} - \mathbf{G}_k)^T = \mathbf{UDV}^T, \qquad \text{for} \quad \mathbf{M} \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}\} \\
\mathbf{R}_{jk} &= \mathbf{VU}^T \\
\mathbf{t}_{jk} &= -\mathbf{R}_{jk}\mathbf{G}_j + \mathbf{G}_k
\end{aligned}$$

Using this comparison of calibration object world points, shown in Figure 3.3, each node now has a local estimate of the relative poses between itself and its four neighbours that comprise its local cluster.

3.5.3 Cluster-based bundle adjustment

Each node j has estimates for the poses of all nodes $k \in \mathcal{N}_j$ relative to itself, which we now refine using bundle adjustment. Like the local calibration stage, the number of parameters to be optimised is minimised by utilising the structure of the calibration object. In addition to the three parameters for the fixed point estimated at node j, $\mathbf{A}_j$, the moving points $\mathbf{B}_{ij}$ and $\mathbf{C}_{ij}$ are again parameterised by spherical coordinates $\theta_i j$ and $\phi_i j$ as in Equation 3.11. Also, the poses require three translation variables and three Euclidean angles. Therefore, the number of parameters to be optimised for each cluster is $3 + 2N + 6d_j$, where d_j is the number of neighbours of node j. This cluster-based bundle adjustment is given by

$$\begin{aligned}
\{\mathbf{R}^*_{jk}, \mathbf{t}^*_{jk}, \mathbf{A}^*_j, \theta^*_{ij}, \phi^*_{ij}\} = \arg\min \sum_{k \in \mathcal{N}_j \cup \{j\}} \sum_{i=0}^{N-1} \frac{1}{2}(||\mathbf{a}_{ik} - \pi_A(\mathbf{A}_j; \mathbf{R}_{jk}, \mathbf{t}_{jk}, \mathbf{K}_k)||_2^2 \\
+ ||\mathbf{b}_{ik} - \pi_B(\theta_{ij}, \phi_{ij}; \mathbf{R}_{jk}, \mathbf{t}_{jk}, \mathbf{K}_k)||_2^2 + ||\mathbf{c}_{ik} - \pi_C(\theta_{ij}, \phi_{ij}; \mathbf{R}_{jk}, \mathbf{t}_{jk}, \mathbf{K}_k)||_2^2).
\end{aligned}$$

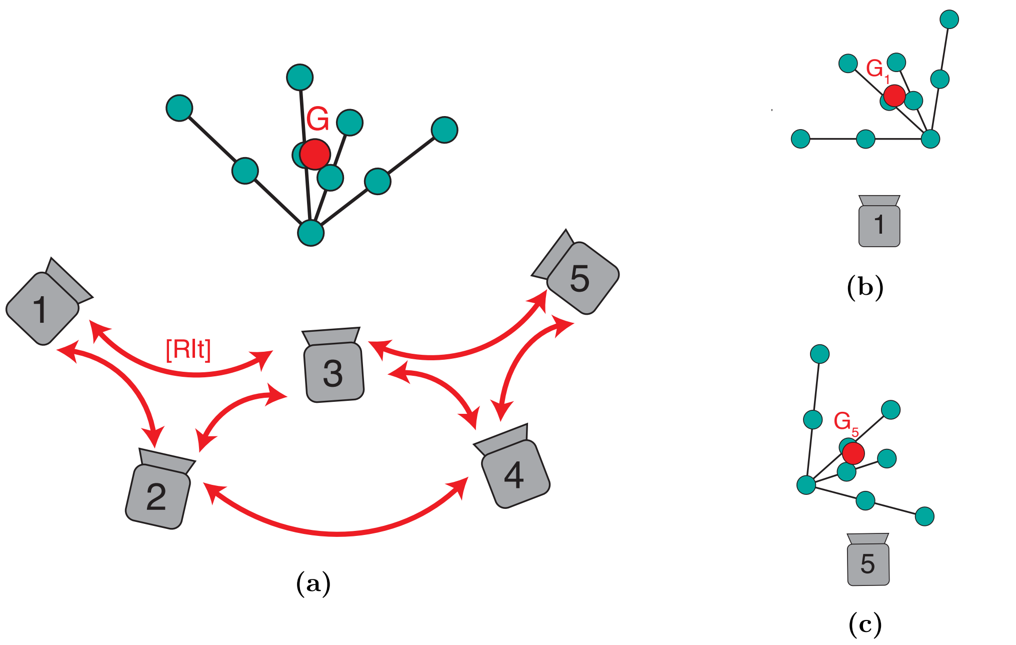


Figure 3.3: The example network of Figure 3.2 estimating extrinsic parameters between neighbours by comparing calibration object observations and the centre of gravity $\mathbf{G}$. A top-level view is given in (a), with the point-of-view of Camera 1 shown in (b) and Camera 5 shown in (c).

It is important to note that each node is operating on slightly different data: the estimations of the fixed point and spherical coordinates are based on the locally acquired image points, and therefore, each local bundle adjustment will arrive at similar but different values. It is these values that need to be brought into global consensus.

During this stage, we also estimated the covariance, $\mathbf{\Sigma}_M$, that will be required by the Gaussian belief propagation. Using covariance estimates for the input image point data, $\mathbf{\Sigma}_m$, we propagated this through the bundle adjustment Jacobian $\mathbf{J}$ with mean-squared error E_{MS} using Equation 3.15 where $(.)^+$ is the Moore–Penrose pseudo-inverse.

$$\mathbf{\Sigma}_M = (E_{MS}(\mathbf{J}^T \mathbf{\Sigma}_m^{-1} \mathbf{J}))^+ \tag{3.15}$$

3.6 Gaussian belief propagation

We now model the true state of each node j as a random vector $\mathbf{z}_j$, containing the extrinsic parameters of the local node and its cluster $k \in \mathcal{N}_j$, producing a $6(d_j + 1)$ vector. $\mathbf{x}_j$ is the noisy observation of $\mathbf{z}_j$ obtained from the bundle adjustment. We wish to estimate $\mathbf{z}_j$ from all $\mathbf{x}_j$ by marginalising the joint density

$$p(\mathbf{z}_j | \mathbf{x}_0, ..., \mathbf{x}_{M-1}) = \int_{(\mathbf{z}_k, k \neq j)} p(\mathbf{z}_0, ..., \mathbf{z}_{M-1} | \mathbf{x}_0, ..., \mathbf{x}_{M-1}) d\mathbf{z}_k.$$

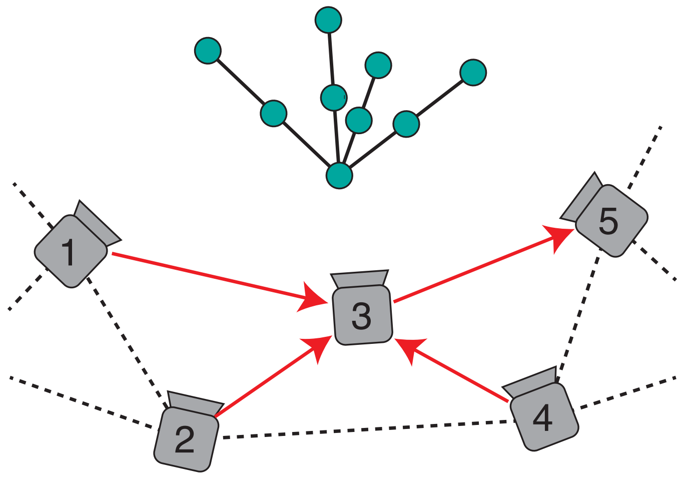


Figure 3.4: The message propagation process described in Figure 2.26 shown for the example distributed network of Figures 3.2 and 3.3. An example message is sent from Camera 3 to Camera 5 containing information that Camera 3 received in the previous iteration from Cameras 1, 2, and 4.

This joint density can be factorised into node potentials ϕ_j and ψ_j,

$$p(\mathbf{z}_0, ..., \mathbf{z}_{M-1}) \propto \prod_{j \in \mathcal{V}} \phi_j(\mathbf{z}_j) \prod_{(j,k) \in \mathcal{E}} \psi_{jk}(\mathbf{z}_j, \mathbf{z}_k).$$

This problem can be solved using the belief propagation algorithm that was presented in Section 2.5.2. Belief propagation is the name given to Pearl's message-passing algorithm which can perform inference on graphs [197]. This algorithm provides exact inference on trees, although it still performs accurately on loopy graphs [200]. It is an iterative algorithm where messages $m^{(t)}_{j \to k}(\mathbf{z}_k)$ are sent from node j to node k across edges at every iteration t.

$$m^{(t)}_{j \to k}(\mathbf{z}_k) \propto \int_{\mathbf{z}_j} \phi_j(\mathbf{z}_j)\psi_{jk}(\mathbf{z}_j, \mathbf{z}_k) \prod_{l \in \mathcal{N}_j \setminus k} m^{(t-1)}_{l \to j}(\mathbf{z}_j) d\mathbf{z}_j \tag{3.16}$$

Here, $\mathcal{N}_j \setminus k$ means all neighbours of j except k. That is, the outgoing messages are the product of incoming messages from the last iteration, excluding that from the message target, combined with the potentials and marginalised over variables not common to both nodes. Figure 3.4 shows this process visually for the camera network, as an extension to Figure 2.26 in Section 2.5.2, showing that all messages received at the sending node, except from the target node, are incorporated into a new message.

The belief at iteration t is similar in construction to the messages, however, it is simply the product of the node potential and all messages from that iteration.

$$b_j^{(t)}(\mathbf{z}_j) \propto \phi_j(\mathbf{z}_j) \prod_{k \in \mathcal{N}_j} m_{k \to j}^{(t)}(\mathbf{z}_j)$$

In *Gaussian* belief propagation, the random variables are assumed to be Gaussian densities, with node potentials, messages and beliefs modelled as two scalars – mean and inverse-variance. The node potential mean, μ_{jj}, is the noisy measurement from the bundle adjustment while the node potential inverse-variance, P_{jj}, is the diagonal of the covariance. In our algorithm, the edge potentials are selector functions which only select the common variables between the two nodes. The messages from Equation 3.16 now have the form

$$\begin{aligned} P_{j \to k}^{(t)} &= P_{jj} + \sum_{l \in \mathcal{N}_j \backslash k} P_{l \to j}^{(t-1)} \\ \mu_{j \to k}^{(t)} &= (P_{jj}\mu_{jj} + \sum_{l \in \mathcal{N}_j \backslash k} P_{l \to j}^{(t-1)} \mu_{l \to j}^{(t-1)}) / P_{j \to k}^{(t)} \end{aligned}$$

and the belief has the form

$$\begin{aligned} P_j^{(t)} &= P_{jj} + \sum_{k \in \mathcal{N}_j} P_{k \to j}^{(t-1)} \\ \mu_j^{(t)} &= (P_{jj}\mu_{jj} + \sum_{k \in \mathcal{N}_j} P_{k \to j}^{(t-1)} \mu_{k \to j}^{(t-1)}) / P_j^{(t)}. \end{aligned}$$

Once the belief means at each node converge, we take this as the estimate for all $\mathbf{z}_j$ and thus the global consensus for the localisation.

3.6.1 Frame alignment

The nodes initially have the pose estimates for their cluster in their own coordinate frames, and therefore cannot converge to a meaningful value. As such, we need all pose measurements to be aligned. To do this we give each node an arbitrary unique identifier. The lowest numbered node is taken as the basis node, b, and all pose measurements of its neighbours, $j \in \mathcal{N}_b$ are aligned to its frame, using node b's estimate of j's pose. As each node becomes aligned to the basis, this is propagated to its unaligned neighbours.

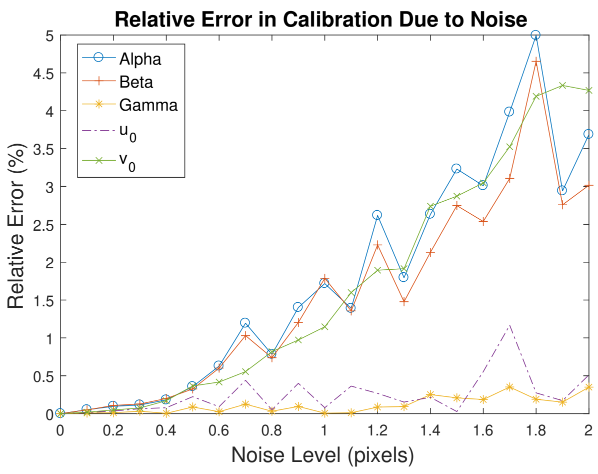


Figure 3.5: Relative error of intrinsic parameter estimation for one-dimensional calibration across Gaussian noise levels up to 2 pixels.

3.7 Experiments and results

In this section, firstly a centralised variant of the algorithm is tested on simulated data to evaluate the performance on the underlying calibration algorithm, then this is compared to the fully distributed algorithm to verify that solving the problem in this way does not degrade the accuracy, and finally real images are used to compare the results against alternative algorithms.

3.7.1 Simulating the one-dimensional calibration

We first simulated a camera with scale factors $\alpha = \beta = 655$px, zero skew, and principal point at the centre of the 1024×768px image. The calibration object was simulated to be 20cm in length with the fixed point located at $\mathbf{A} = [0\ 0\ 75]^T$cm. Image points for the object were then generated at 100 different orientations by sampling θ in $[0, \frac{\pi}{4}]$ and ϕ in $[0, 2\pi]$ with uniform distribution. We calculated the average error across 200 independent trials for each level of Gaussian noise with zero mean and σ between 0.1 to 2.0 pixels. We measured the relative error for all parameters with respect to α, giving $|\Delta\alpha/\alpha|$, $|\Delta\beta/\alpha|$, $|\Delta\gamma/\alpha|$, $|\Delta u_0/\alpha|$, $|\Delta v_0/\alpha|$, as proposed by Triggs [244]. The results are shown in Figure 3.5, with errors staying below 5% at two pixels of noise. Calibration object detection from a real camera generally show less than one pixel of noise, and therefore should perform at below 2% error for intrinsic parameters.

3.7.2 Simulating the distributed localisation

Next, we performed a simulation of the full algorithm. In this experiment, we used the same intrinsic parameters as previous and similarly used a 20cm calibration

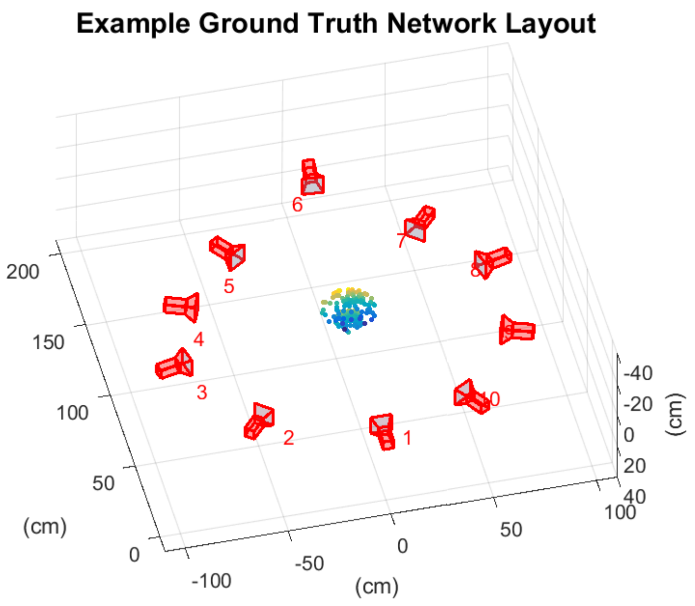
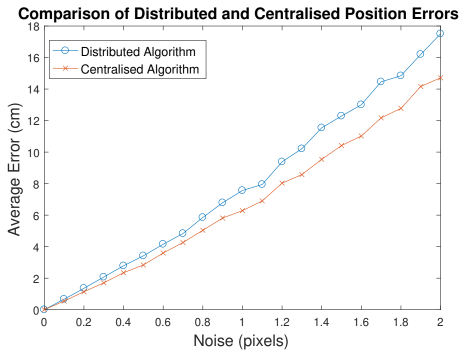


Figure 3.6: The ground truth layout for one trial of our simulation, using ten cameras nodes.

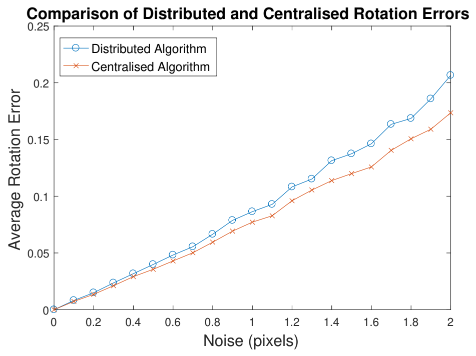
object with $\mathbf{A} = [0\ 0\ 75]^T$cm uniformly sampled 100 times for each trial from θ in $[0, \frac{\pi}{4}]$ and ϕ in $[0, 2\pi]$. Our cameras were located in a mostly circular fashion on the xy-plane around the fixed point and spaced $2\pi/M$ apart at distances uniformly sampled from $[75, 95]$cm. For our experiment, we used the number of camera nodes as $M = 10$. The cameras were initially directed towards the fixed point but had their y-axis rotations perturbed by a uniform sampling of $[-5°, 5°]$. Once again, we took the average error of 200 trials at each noise level from 0.1 to 2.0 pixels. An example ground truth layout from our simulation is shown in Figure 3.6, with the cameras shown located about the world points. In these simulations, our method of constructing the vision graph resulted in each node being connected to its two neighbours either side.

In our simulation, we first performed the local intrinsic calibration and used the estimates of the calibration object to construct our vision graph. We then performed cluster-based bundle adjustment followed by Gaussian belief propagation until convergence. Our criteria for convergence was that the proportional change in belief $|b_j^t - b_j^{t-1}|/b_j^t$ was less than 0.0001 for all nodes. We found that convergence occurred in between 12 and 40 iterations for all trials.

The error metric we used for the localisation was that employed by Devarajan and Radke [156], using the error in position as $||\Delta C||$ and error in orientation as $2\sqrt{1 - \cos\Delta\theta}$, where $\Delta\theta$ is the relative rotation between the ground truth and estimated value. We averaged these errors across all nodes and trials for each noise level and compared the values to the errors of a centralised global bundle adjustment. The position errors in the distributed and centralised algorithms can be seen



(a) The position error for noise levels using the distributed and centralised algorithms.



(b) The rotation error for noise level using the distributed and centralised algorithms.

Figure 3.7: The error between the distributed algorithm for optimising and align localisation compared to a centralised global bundle adjustment, shown across Gaussian noise levels up to 2 pixels.

in Figure 3.7a, while the equivalent rotations errors can be seen in Figure 3.7b.

As can be seen from Figure 3.7, the centralised algorithm only performed slightly better than the distributed algorithm across all noise levels. The reason for the difference is that the centralised algorithm is optimising over a joint global objective, whereas the distributed algorithm optimises over a number of different local sub-objectives that are then brought into consensus with essentially a variance-weighted distributed average. This means that the two overall objectives are not completely equivalent. The results, however, show that the benefits of the distributed algorithm, such as being resistant to single node failure and being scalable without causing communication bottlenecks, are achievable without substantial degradation to the quality of the localisation. Furthermore, at up to two pixels of noise the error in both translation and rotation gradually increases, showing no clear point of sudden failure. This highlights that the algorithm remains reliable in noisy environments.

3.7.3 Evaluation on real data

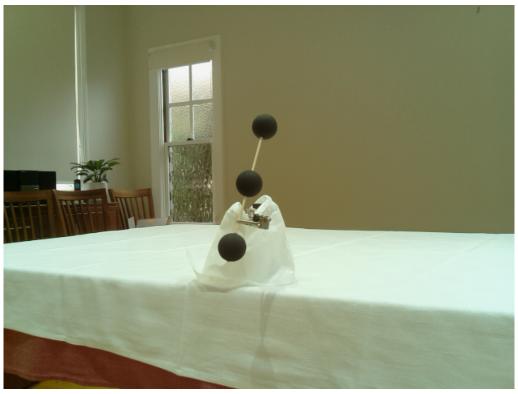
Finally, the performance of the algorithm was verified on real data using a network of ten Raspberry Pi 3 computers with camera modules. The 'ideal' intrinsic parameters were taken to be those from the respective data-sheets as given in Table 3.1. Although these are not exact ground truth values, modern cameras are manufactured to a high enough standard that this provides a suitable baseline. Due to this lack of a ground truth, the root-mean-squared reprojection error (RMSE) was also compared between the image points and the reconstructed world points. The cameras were arranged in an arc, with their centre distances and rotations relative to camera 0 reported in Table 3.2. The calibration object was 205mm long, with three equidistant 40mm balls used as points. The dataset contained 30 images of the

Table 3.1: Ideal intrinsics from camera data-sheets (in pixels).

Camera	α	β	c_u	c_v
Cameras 0-4	634	634	320	240
Cameras 5-7	612	612	320	240
Cameras 8-9	501	501	320	240

Table 3.2: Ground truth for centre distances and y-axis rotations, relative to Camera 1.

Camera	1	2	3	4	5	6	7	8	9
Distance (mm) ± 0.5mm	134	263	399	519	638	752	895	1064	1228
Rotation (deg) $\pm 0.5^\circ$	9	14	20	26	31	40	48	48	62


(a) 1D calibration object

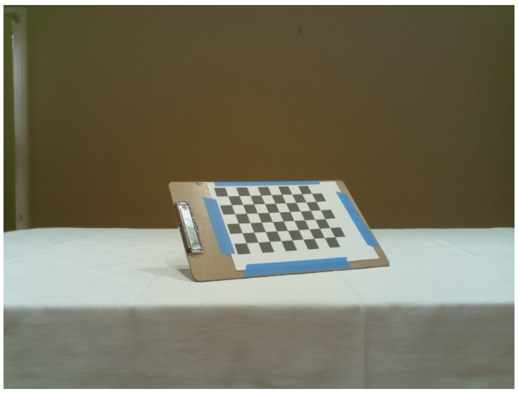

(b) 2D calibration object

Figure 3.8: Example images from the experimental test showing the scene with a 1D and 2D calibration object.

calibration object where it was visible to all ten cameras, with the vision graph constructed using the two nearest neighbours either side as determined by the method of Section 3.5.1.

The intrinsic and extrinsic parameters were calculated firstly using the 1D algorithm without the improvements of Section 3.4.2, then the improved centralised algorithm, as well as the distributed version of the algorithm. This was also compared to the 2D calibration of Zhang [13]. Example images from the dataset for each calibration type are shown in Figure 3.8, with results shown in Table 3.3.

As can be seen from the results in Table 3.3, the original 1D calibration algorithm does not perform as well as the ubiquitous 2D calibration algorithm, however, the improvements discussed in Section 3.4.2 bring these two types of calibration more in-line with each other. Furthermore, extending the 1D algorithm to its distributed form maintains a similar accuracy to the centralised version. This demonstrates that the distributed 1D calibration algorithm achieves the goal of providing a simple calibration algorithm with comparable quality to the most common alternative whilst

Table 3.3: Relative difference to ground truth and ideal results averaged per algorithm, and RMS reprojection error.

Algorithm	α	β	c_u	c_v	T	R_y	RMSE
2D [13]	0.63%	0.75%	1.15%	1.02%	2.65%	2.90%	0.126px
1D [15]	2.24%	1.04%	1.99%	2.74%	4.20%	3.25%	0.572px
1D Improved	0.89%	0.93%	1.51%	2.56%	4.03%	3.11%	0.481px
1D Distributed	1.07%	1.04%	1.59%	2.78%	4.21%	3.15%	0.486px

being usable at wide angles in a distributed network. The performance of the algorithm is comparable to the simulation results in the range of $0.5 - -1$ pixel of noise, demonstrating that there is still room for adequate performance in camera systems affected by even higher levels of noise.

3.8 Summary

In this chapter, we have presented a distributed algorithm for camera sensor network calibration and localisation with known scale. The motivation for this was to provide a robust initialisation stage for the distributed calibration, localisation, and mapping pipeline presented in Figures 1.2 and 2.28. The design goal for this initialisation stage was to use a calibration method that is able to determine the scale of the scene without introducing issues of calibration object self-occlusion. Therefore, one-dimensional calibration was chosen.

Our method models the network as a Markov random field based on a vision graph. Following the one-dimensional calibration performed locally at each node, a vision graph was constructed based on rotation distance between nodes. Then, the relative extrinsic parameters between neighbouring nodes were estimated from the comparison between estimated calibration object world points and were optimised locally with cluster-based bundle adjustment. These estimates are brought to a globally accurate consensus through Gaussian belief propagation on the vision graph. Our method was shown to be comparable to centralised calibration and bundle adjustment whilst having the advantages of distributed algorithms, namely resistance to node failure and scalability. Although the widely-used 2D calibration has higher accuracy in its result, we have brought the 1D calibration method closer in performance whilst being in a distributed setting where the use of the 2D method is less suitable.

From this, it can be seen that a suitable initialisation stage for the distributed robotic vision pipeline has been proposed.

Chapter 4

Robust one-dimensional calibration and localisation of a distributed camera sensor network[a]

Contents

4.1 Summary of contributions

- An improved version of the distributed one-dimensional calibration and localisation algorithm is proposed, using general-motion one-dimensional calibration and distributed optimisation.
- Normalisation is applied to the projective 3D reconstructions in general-motion one-dimensional camera calibration to improve the numerical stability of the linear estimation.

[a]Chapter adapted from "B. Halloran *et al.*, 'Robust one-dimensional calibration and localisation of a distributed camera sensor network,' *Pattern Recognition*, vol. 98, 107058, pp. 1–12, 2020. DOI: `10.1016/j.patcog.2019.107058`"

- A more geometrically meaningful bundle adjustment is introduced to replace the multiple stages of nonlinear optimisation in the original method, improving accuracy and computational efficiency.
- Due to the general-motion of the calibration object, the vision graph is initialised from covisibility of the object, rather then the lattice graph used previously.
- The global bundle adjustment objective of the centralised algorithm is split into local objectives at each node using alternating direction method of multipliers which are brought into global consensus using Gaussian belief propagation.
- Extensive testing is done on synthetic and real data.
- Results show that the centralised algorithm has superior performance to the original algorithm, self-calibration, and 2D calibration for a number of intrinsic parameters and the extrinsic localisation.
- The distributed algorithm achieves similar accuracy to the centralised algorithm for all parameters.

4.2 Introduction

This chapter explores an improved version of the calibration and localisation method discussed in Chapter 3, providing a better initialisation stage for the three-stage distributed robotic vision pipeline presented in Figures 1.2 and 2.28, which has the overall goals of accurate calibration, localisation, and mapping.

Calibration and localisation of a camera sensor network (CSN) enables higher-level computer vision tasks by allowing metric 3D information to be attained from 2D images. Most multi-view calibration algorithms use a central processor with access to data from all camera nodes, which can encounter communication issues in large CSNs and represents a single point of failure. With low-cost on-board processing becoming increasingly feasible for robotic vision applications, such as networks of *micro aerial vehicles*, distributed processing can ensure robustness and scalability. This promotes the need for a simple calibration and localisation algorithm for distributed CSNs.

The most popular calibration algorithm is Zhang's '2D' calibration using a planar pattern seen at arbitrary orientations [13] which works on various camera models [245, 246]. Unfortunately, in a large CSN, the pattern self-occludes when viewed at wide angles. Alternatively, 'self-calibration' doesn't use any pattern, instead using structure from motion (SfM) [17, 79]. While suited for large CSNs, it is computationally complex and cannot determine scene scale. This chapter utilises '1D' calibration [15], where collinear points are seen at arbitrary orientations. This work also uses Gaussian belief propagation (GaBP), a message passing algorithm for aligning local estimates in a distributed network [200], and alternating direction method of multipliers (ADMM), a method of splitting optimisation problems for distributed processing [215].

These features provide an improved distributed calibration and localisation algorithm that can accurately initialise a robotic vision platform, resulting in a better initialisation stage for the pipeline discussed in this dissertation.

4.2.1 Related work

The original 1D calibration algorithm of Zhang [15], as utilised in Chapter 3, was described for a single camera and constrained the calibration object to rotate about a fixed point, however, research has relaxed this to allow for planar motion and motion under gravity [70]. Accuracy and speed of this algorithm has also been improved using normalised image points and a partitioned Levenberg-Marquardt refinement [72], information weighted algorithms [74], and replacing non-linear optimisation with convex relaxation [243]. Full general motion of the calibration object has been achieved in multi-view CSNs, originally requiring a reference camera that was already calibrated [85], and later without this requirement based on vanishing points [86], robust perspective factorisation [247], and then fundamental matrices [87]. Our method improves the accuracy of the latter algorithm using two stages of normalisation and a single global bundle adjustment (BA).

Our method also utilises *belief propagation*, originally for inference on trees [197], and later extended to loopy graphs [198] and continuous variables [200]. Most distributed calibration algorithms use self-calibration, with Devarajan et al. modelling the CSN as a Markov random field (MRF) where each node performs local SfM which is aligned with belief propagation [7]. Tron and Vidal brought pairwise CSN pose estimates to alignment with average consensus-based gradient descent on Riemannian manifolds [8]. There is also a range of work on distributed localisation for dynamic robots under changing topologies using methods such as dynamic average consensus [248], fusion of relative measurements [249], robust control laws [250], and Kalman filter-based collaborative localisation [251]. Furthermore, Eriksson et al. have demonstrated splitting BA across processors using proximal splitting, similar to ADMM [223]. Graphical models have wide use in BA [252] and localisation [253], and similar to the distributed case need to ensure consistent poses along graph cycles [254]. Our method uses a combination of ADMM and GaBP to optimise the localisation over an MRF.

4.2.2 Contributions

Our motivation was to develop a simple and robust calibration technique for ad-hoc CSNs, particularly for inexpensive *micro aerial vehicles* that might be replaced or swapped often, where it could be easily run to provide a quick initialisation to a known scale for a dynamic localisation algorithm such as in [249]. General motion 1D calibration is well suited to a CSN due to the lack of self-occlusion at wide angles, however, such algorithms don't yet have comparable accuracy to other calibration algorithms. Furthermore, distributed processing is being used to ensure the algorithm is highly scalable. From this motivation, the primary goal of this chapter is to produce a robust initialisation stage for the pipeline presented in Figures 1.2 and 2.28.

As a basis for our work we have used the calibration and localisation algorithm of de França [87]. However, we have used normalised image points and projective 3D reconstructions to improve the accuracy of key linear estimations. Then, we have introduced a more geometrically meaningful non-linear refinement to replace all non-linear refinement of the previous method. Next, to adapt this algorithm to a distributed network for alignment with ADMM and GaBP we have taken additional steps. We have operated on neighbourhood-based node clusters to perform the calibration separately at each camera. We then used ADMM to split the global bundle adjustment objective into locally-computable steps. Finally, we align each iteration of the ADMM process using an application of GaBP that considers frame alignment and an error model.

The chapter is organised as follows: Section 4.3 introduces the preliminaries of the camera model, calibration object, and distributed network. Section 4.4 describes the 1D calibration algorithm and our improvements, and Section 4.5 adapts this to a distributed network. Section 4.6 provides experimental results on simulated and real data, and Section 4.7 concludes the chapter.

4.3 Preliminaries

In Section 2.3.1 the camera calibration problem was introduced and in Section 2.5.1 the representation of a CSN as two undirected graphs was introduced. However, in the following section, the important aspects of these preliminary concepts are reintroduced for clarity.

4.3.1 The pinhole camera model

Different to Section 3.3, this chapter expresses the multi-view pinhole model more explicitly and discusses greater detail regarding the parameterisation of poses. Furthermore, projective reconstructions are utilised in this chapter.

Consider a network of $K+1 \geq 2$ camera nodes with node 0 at the origin of the world coordinate system. Each node k has an intrinsic matrix $\mathbf{K}_k$ and extrinsic matrix $[\mathbf{R}_k|\mathbf{t}_k]$, and takes N images of M points, $\mathbf{m} = [x, y]^T$, a projection of the 3D world point $\mathbf{M} = [X, Y, Z]^T$. These have homogeneous forms $\tilde{\mathbf{m}} = [x, y, 1]^T$ and $\tilde{\mathbf{M}} = [X, Y, Z, 1]^T$ respectively. The jth image point of the ith image for node k is related to its corresponding world point by

$$\tilde{\mathbf{m}}_{kji} \simeq \mathbf{K}_k[\mathbf{R}_k|\mathbf{t}_k]\tilde{\mathbf{M}}_{ji}, \qquad \textit{where} \quad \mathbf{K}_k = \begin{bmatrix} \alpha_k & 0 & u_k \\ 0 & \beta_k & v_k \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.1}$$

Here, $\simeq$ signifies equality to a scale factor. Extrinsic parameters $\mathbf{R}_k$ and $\mathbf{t}_k$ are rotation and translation of the camera, with $[\mathbf{R}_0|\mathbf{t}_0] = [\mathbf{I}|\mathbf{0}]$. We have x- and y-axis focal lengths α_k and β_k, and (u_k, v_k) as the image principal point. The combined camera projection matrix is $\mathbf{P}$,

$$\mathbf{P}_k = \mathbf{K}_k[\mathbf{R}_k|\mathbf{t}_k], \qquad \mathbf{P}_0 = \mathbf{K}_0[\mathbf{I}|\mathbf{0}]. \tag{4.2}$$

We use the minimal Lie-algebra representation of $[\mathbf{R}|\mathbf{t}]$ with $\xi = (\omega, \nu)^T \in \mathfrak{se}(3)$, related by the exponential map of Equation 4.3 with corresponding inverse mapping $\log_{SE(3)}(\cdot)$, and use $[\mathbf{v}]_\times$ for the cross-product matrix $[\mathbf{v}]_\times \mathbf{x} = \mathbf{v} \times \mathbf{x}$.

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \exp_{\mathfrak{se}(3)}(\hat{\xi}) \tag{4.3}$$
$$\textit{where} \quad \hat{\xi} = \begin{bmatrix} [\omega]_\times & \nu \\ \mathbf{0} & 0 \end{bmatrix}$$

The correspondences $(\mathbf{m}_{0ji} \leftrightarrow \mathbf{m}_{kji})$ for nodes 0 and k can be used to find the fundamental matrix $\mathbf{F}_k$, epipole $\mathbf{e}_k$, and projective matrices, $\mathcal{P}_0$ and $\mathcal{P}_k$, which are related to the camera matrices of Equation 4.2 by a transform $\mathbf{T}_k$ [19].

$$\mathcal{P}_0 = [\mathbf{I}|\mathbf{0}] = \mathbf{P}_0\mathbf{T}_k, \qquad \mathcal{P}_k = [\mathbf{H}|\tilde{\mathbf{e}}_k] = \mathbf{P}_k\mathbf{T}_k \tag{4.4}$$
$$\textit{where} \quad \mathbf{H} = [\tilde{\mathbf{e}}_k]_\times \mathbf{F}_k$$
$$\mathbf{T}_k = \mu_k \begin{bmatrix} \mathbf{K}_0^{-1} & \mathbf{0} \\ \mathbf{w}_k^T & w_k \end{bmatrix} \tag{4.5}$$

Here, μ_k is a scale factor and $\mathbf{W}_k = [\mathbf{w}_k^T \; w_k]^T$ is the plane at infinity for the projective coordinate system of this pair. Triangulating [19] the $(\mathbf{m}_{0ji} \leftrightarrow \mathbf{m}_{kji})$ matches with $\mathcal{P}_0$ and $\mathcal{P}_k$ produces projective reconstructions, $\mathcal{M}_{kji}$

$$\tilde{\mathbf{m}}_{kji} \simeq \mathcal{P}_k\tilde{\mathcal{M}}_{kji}, \qquad \textit{and} \qquad \tilde{\mathbf{m}}_{0ji} \simeq \mathcal{P}_0\tilde{\mathcal{M}}_{kji}. \tag{4.6}$$

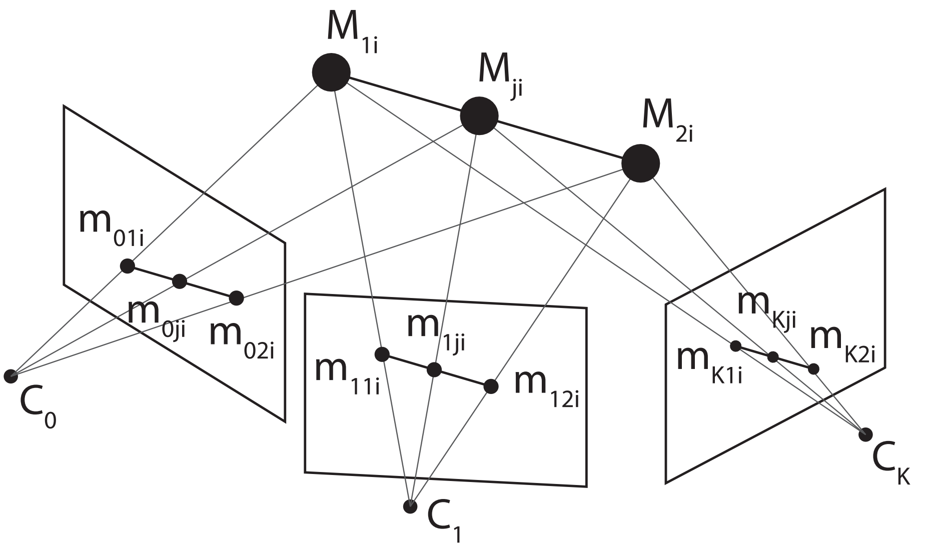


Figure 4.1: The general-motion one-dimensional calibration object as it appears to various camera nodes.

4.3.2 The one dimensional calibration object

The 1D calibration object, used to determine camera matrices $\mathbf{P}_k$, consists of $M \geq 3$ collinear points seen at $N \geq 5$ arbitrary displacements by each camera, as shown in Figure 4.1, providing each node with MN image points. This is contrasted with the structure presented in Figure 3.1 in Chapter 3, where the calibration object required a fixed point. The cameras require overlapping fields of view such that the M points of the calibration object are present in at least 5 images in common with camera 0. The known object length and ratios between points is given in Equations 4.7–4.8.

$$L = ||\mathbf{M}_{1i} - \mathbf{M}_{2i}|| \tag{4.7}$$

$$\mathbf{M}_{ji} = \lambda_{1j}\mathbf{M}_{1i} + \lambda_{2j}\mathbf{M}_{2i}, \qquad for \quad 3 \leq j \leq M. \tag{4.8}$$

4.3.3 Camera sensor networks as Markov random fields

As in Chapter 3, the ad-hoc network of $K + 1$ camera nodes is described by an undirected vision graph, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{0, ..., K\}$ is the set of camera nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges with $(i, j) \in \mathcal{E}$ pairs of nodes with overlapping fields of view. We denote the direct neighbours of node i as $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$, with degree $d_i = |\mathcal{N}_i|$. Figure 2.24 in Section 2.5.1 demonstrates the relationship between the fields of view and the vision graph.

4.4 Multi-view calibration with one-dimensional objects

In this section we briefly describe the multi-view 1D calibration algorithm of de França, then we detail our three improvements to this method. We present a flowchart of the full algorithm in Figure 4.2.

4.4.1 De França's one dimensional calibration with general motion

Firstly, we describe the three linear least-squares (LLS) problems and two non-linear refinements of the original algorithm. The first two LLS problems are run for nodes 0 and k, whilst the third is run for all remaining nodes. For the original derivation we refer the reader to [87].

The first projective plane at infinity

The first LLS system finds the projective plane at infinity between nodes 0 and k, $\mathbf{W}_k$, using Equation 4.9, which is stacked $(M-2)N$ times for the system of linear equations in Equation 4.10. Here, the inputs are the projective reconstructions, $\mathcal{M}_{kji}$, as well as the ratio of lengths between world points along the calibration object, λ_{1j} and λ_{2j} for $3 \leq j \leq M$, which is solved for $\mathbf{W}_k$.

$$\mathbf{u}_{kji}^T \mathbf{W}_k = 0 \tag{4.9}$$

$$where \quad \mathbf{u}_{kji}^T = \tilde{\mathcal{M}}_{k1i} + \frac{\lambda_{1j}(\mathcal{M}_{k1i} \times \mathcal{M}_{kji}).(\mathcal{M}_{k2i} \times \mathcal{M}_{kji})}{\lambda_{2j}(\mathcal{M}_{k2i} \times \mathcal{M}_{kji}).(\mathcal{M}_{k2i} \times \mathcal{M}_{kji})} \tilde{\mathcal{M}}_{k2i}.$$

$$\mathbf{U}_k^T \mathbf{W}_k = 0 \tag{4.10}$$

$$where \quad \mathbf{U}_k = \begin{bmatrix} \mathbf{u}_{k30}^T & \mathbf{u}_{k31}^T & ... & \mathbf{u}_{kji}^T & ... & \mathbf{u}_{kMN}^T \end{bmatrix}$$

Image of the absolute conic

The second LLS problem relates the plane at infinity to the image of the absolute conic (IAC), $\omega_k = \mu_k^2 \mathbf{K}_0^{-T} \mathbf{K}_0^{-1}$, where μ_k is a homogeneous scale factor. This is a 3×3 symmetric matrix that is represented in a vector form, $\mathbf{\Omega}_k =$

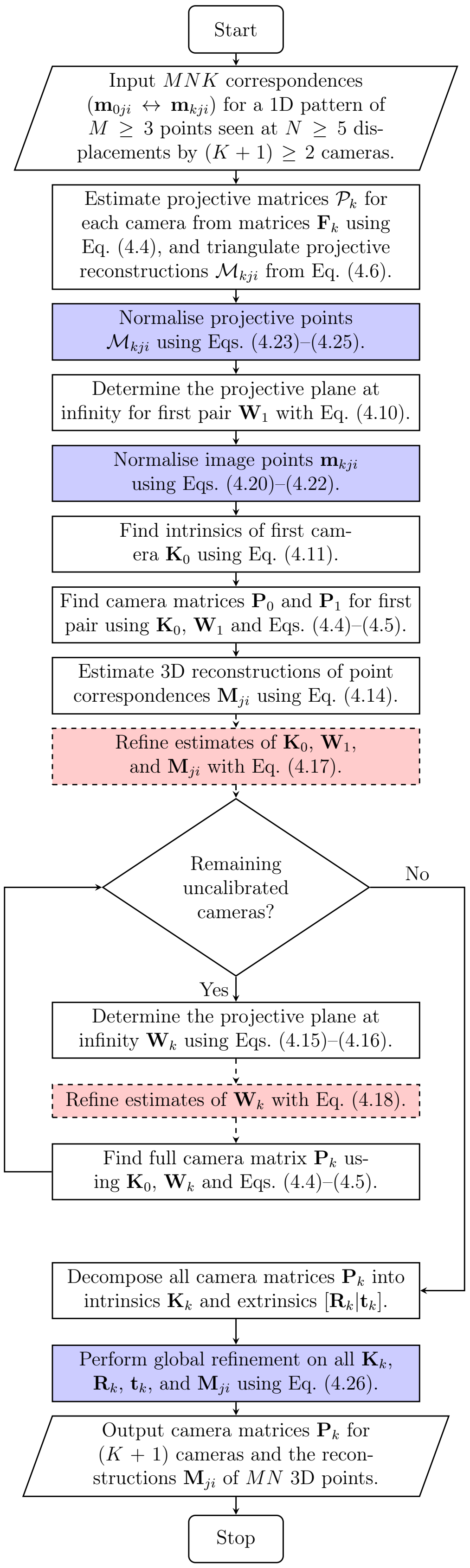

Start
Input MNK correspondences ($\mathbf{m}_{0ji} \leftrightarrow \mathbf{m}_{kji}$) for a 1D pattern of $M \geq 3$ points seen at $N \geq 5$ displacements by $(K+1) \geq 2$ cameras.
Estimate projective matrices $\mathcal{P}_k$ for each camera from matrices $\mathbf{F}_k$ using Eq. (4.4), and triangulate projective reconstructions $\mathcal{M}_{kji}$ from Eq. (4.6).
Normalise projective points $\mathcal{M}_{kji}$ using Eqs. (4.23)–(4.25).
Determine the projective plane at infinity for first pair $\mathbf{W}_1$ with Eq. (4.10).
Normalise image points $\mathbf{m}_{kji}$ using Eqs. (4.20)–(4.22).
Find intrinsics of first camera $\mathbf{K}_0$ using Eq. (4.11).
Find camera matrices $\mathbf{P}_0$ and $\mathbf{P}_1$ for first pair using $\mathbf{K}_0$, $\mathbf{W}_1$ and Eqs. (4.4)–(4.5).
Estimate 3D reconstructions of point correspondences $\mathbf{M}_{ji}$ using Eq. (4.14).
Refine estimates of $\mathbf{K}_0$, $\mathbf{W}_1$, and $\mathbf{M}_{ji}$ with Eq. (4.17).
Remaining uncalibrated cameras?
No
Yes
Determine the projective plane at infinity $\mathbf{W}_k$ using Eqs. (4.15)–(4.16).
Refine estimates of $\mathbf{W}_k$ with Eq. (4.18).
Find full camera matrix $\mathbf{P}_k$ using $\mathbf{K}_0$, $\mathbf{W}_k$ and Eqs. (4.4)–(4.5).
Decompose all camera matrices $\mathbf{P}_k$ into intrinsics $\mathbf{K}_k$ and extrinsics $[\mathbf{R}_k|\mathbf{t}_k]$.
Perform global refinement on all $\mathbf{K}_k$, $\mathbf{R}_k$, $\mathbf{t}_k$, and $\mathbf{M}_{ji}$ using Eq. (4.26).
Output camera matrices $\mathbf{P}_k$ for $(K+1)$ cameras and the reconstructions $\mathbf{M}_{ji}$ of MN 3D points.
Stop

Figure 4.2: Improved calibration procedure, with our novel additions highlighted in blue and the no-longer needed non-linear refinements steps shown in red with dashed lines.

$\left[\omega_{11}\ \omega_{12}\ \omega_{22}\ \omega_{13}\ \omega_{23}\ \omega_{33}\right]^T$, to solve the $2MN$ linear equations,

$$\left[\mathbf{d}_{k00}\ ...\ \mathbf{d}_{kji}\ ...\ \mathbf{d}_{kMN}\right]^T \boldsymbol{\Omega}_k = \left[\mathbf{L}_{11}\ ...\ \mathbf{L}_{ji}\ ...\ \mathbf{L}_{MN}\right]^T \tag{4.11}$$

$$where \quad \mathbf{d}_{kji} = \begin{bmatrix} g_1^2 & 2g_1g_2 & g_2^2 & 2g_1g_3 & 2g_2g_3 & g_3^2 \\ q_1^2 & 2q_1q_2 & q_2^2 & 2q_1q_3 & 2q_2q_3 & q_3^2 \end{bmatrix}^T \tag{4.12}$$

$$\mathbf{L}_{ji} = \left[\lambda_{1j}^2 L^2 \quad \lambda_{2j}^2 L^2\right]$$

$$\mathbf{g}_{kji} = (\eta_{kji}\tilde{\mathbf{m}}_{0ji} - \eta_{k2i}\tilde{\mathbf{m}}_{02i})$$

$$\mathbf{q}_{kji} = (\eta_{k1i}\tilde{\mathbf{m}}_{01i} - \eta_{kji}\tilde{\mathbf{m}}_{0ji})$$

$$\eta_{kji} = \frac{\mathcal{Z}_{kji}}{\tilde{\mathcal{M}}_{kji}^T \mathbf{W}_k}. \tag{4.13}$$

In Equation 4.12, g_n and q_n are the nth entries in $\mathbf{g}_{kji}$ and $\mathbf{q}_{kji}$, and in Equation 4.13 $\mathcal{Z}_{kji}$ is the depth of $\mathcal{M}_{kji}$, with $\mathcal{M} = [\mathcal{X}\ \mathcal{Y}\ \mathcal{Z}]^T$. The inputs to the LLS problem in Equation 4.11 are the image points, projective reconstructions, calibration object length and $\mathbf{W}_k$, with the output being the IAC which is decomposed into $\mathbf{K}_0$. We can now upgrade our projective matrices and reconstructions to Euclidean camera matrices and world points with Equations 4.4–4.5 and

$$\tilde{\mathbf{M}}_{ij} \simeq \mathbf{T}_k \tilde{\mathcal{M}}_{kji}. \tag{4.14}$$

Remaining projective planes at infinity

The final LLS problem, repeated for all remaining cameras, finds the inverse projective plane at infinity, $\bar{\mathbf{W}}_k = \left[\bar{\mathbf{w}}_k^T \quad \bar{w}_k\right]^T$. We also use $\mathcal{P}_k = [\mathcal{P}_{Ak}|\mathcal{P}_{bk}]$ with left 3×3 matrix $\mathcal{P}_{Ak}$ and right 3-vector $\mathcal{P}_{bk}$.

$$\left[\boldsymbol{\Psi}_{k00}\ ...\ \boldsymbol{\Psi}_{kji}\ ...\ \boldsymbol{\Psi}_{kMN}\right]^T \bar{\mathbf{W}}_k = \left[\psi_{k00}\ ...\ \psi_{kji}\ ...\ \psi_{kMN}\right]^T \tag{4.15}$$

$$where \quad \boldsymbol{\Psi}_{kji} = [\tilde{\mathbf{m}}_{kji}]_\times \mathcal{P}_{bk} \tilde{\mathbf{M}}_{ji}^T$$

$$\psi_{kji} = -[\tilde{\mathbf{m}}_{kji}]_\times \mathcal{P}_{Ak} \mathbf{K}_0 \mathbf{M}_{ji}$$

$$\mathbf{W}_k = \left[(1/\bar{w}_k)\bar{\mathbf{w}}_k^T \mathbf{K}_0^T \quad (1/\bar{w}_k)\right]^T \tag{4.16}$$

In Equation 4.15, the inputs are the image points, projective matrices, Euclidean world points, and intrinsic matrix $\mathbf{K}_0$. Its output, $\bar{\mathbf{W}}_k$, is related to $\mathbf{W}_k$ by Equation 4.16 and used in Equation 4.5 to find the remaining camera matrices.

Non-linear refinement

The original algorithm also had two non-linear least-squares (NLLS) refinements to improve accuracy assuming image point noise; the first refined the result of Equation 4.11 with Equation 4.17 and the second refined the result of Equation 4.15 with Equation 4.18. These are minimised with a curve-fitting algorithm, such as Levenberg-Marquardt [255].

$$\sum_{k=0}^{1}\sum_{i=1}^{N}\sum_{j=1}^{M} ||\mathbf{m}_{kji} - \pi'_{kji}(\mathbf{K}_0,\ \mathbf{W}_1,\ \mathbf{M}_{1i},\ \theta_i,\ \phi_i)||^2 \tag{4.17}$$

$$\sum_{k=2}^{K}\sum_{i=1}^{N}\sum_{j=1}^{M} ||\mathbf{m}_{kji} - \pi''_{kji}(\mathbf{W}_k\ ;\ \mathbf{K}_0,\ \mathbf{M}_{ji})||^2 \tag{4.18}$$

In Equations 4.17–4.18, π'_{kji} and π''_{kji} are the reprojection of $\mathbf{M}_{ji}$ into the image of node k using Equations 4.1–4.2, getting $\mathbf{P}_k$ from $\mathbf{K}_0$ and $\mathbf{W}_k$ with Equations 4.4–4.5. In Equation 4.17 all parameters of π'_{kji} are refined, whereas in Equation 4.18 only $\mathbf{W}_k$ is refined. We reduce the number of parameters in Equation 4.17 by parameterising the collinear calibration points $\mathbf{M}_{ji}$ for $2 \leq j \leq M$ by θ_i and ϕ_i,

$$\mathbf{M}_{ji} = \mathbf{M}_{1i} + \lambda_{2j} L \begin{bmatrix} \sin\theta_i \cos\phi_i \\ \sin\theta_i \sin\phi_i \\ \cos\theta_i \end{bmatrix}. \tag{4.19}$$

4.4.2 Our improvements to the one-dimensional calibration

We have improved the method of Section 4.4.1 by normalising the image points and projective reconstructions for the LLS problems, and replacing the NLLS problems with a global BA. These changes are shown in Figure 4.2.

Normalising the image points and projective peconstruction

Noise in the image point data affects the estimation of the IAC in Equation 4.11. From Hartley [45], a transformation of input data can improve LLS solutions affected by noise, which has seen success in other calibration algorithms [72, 74]. The main constraint between the image points and IAC being exploited by Equation 4.11 is given in Equation 4.20 [19]. Applying some transformation to the 2D points, $\mathbf{S}_{2D}$ in Equation 4.21, we can satisfy this constraint with a transformed IAC, $\hat{\omega}$, corresponding a transformed intrinsic matrix $\hat{\mathbf{K}}_0$, and can recover the original $\mathbf{K}_0$ using 4.22.

$$\tilde{\mathbf{m}}_{0ji}^T \omega \tilde{\mathbf{m}}_{0ji} = 0 \tag{4.20}$$

$$\hat{\mathbf{m}}_{kji} = \mathbf{S}_{2D} \tilde{\mathbf{m}}_{kji} \tag{4.21}$$

$$\hat{\mathbf{m}}_{0ji}^T \mathbf{S}_{2D}^{-T} \omega \mathbf{S}_{2D}^{-1} \hat{\mathbf{m}}_{0ji} = \hat{\mathbf{m}}_{0ji}^T \hat{\omega} \hat{\mathbf{m}}_{0ji} = 0$$

$$\hat{\omega} = \mathbf{S}_{2D}^{-T} \mathbf{K}_0^{-T} \mathbf{K}_0^{-1} \mathbf{S}_{2D}^{-1} = \hat{\mathbf{K}}_0^{-T} \hat{\mathbf{K}}_0^{-1}$$

$$\mathbf{K}_0 = \mathbf{S}_{2D}^{-1} \hat{\mathbf{K}}_0 \tag{4.22}$$

Similarly, the estimation of the projective plane at infinity between $\mathcal{P}_0$ and $\mathcal{P}_k$, $\mathbf{W}_k$, significantly affects estimations of $\mathbf{K}_0$ and $\mathbf{T}_k$, used in the Euclidean reconstructions $\mathbf{M}_{ij}$ and in Equation 4.15. Considering the constraint between $\mathbf{W}_k$ and $\tilde{\mathcal{M}}_{kji}$, given by Equation 4.23 [19], if we transform the projective 3D points with $\mathbf{S}_{3D}$ in Equation 4.24, we continue to satisfy the constraint with the transformed $\hat{\mathbf{W}}_k$, related back to the original $\mathbf{W}_k$ by Equation 4.25.

$$\mathbf{W}_k^T \tilde{\mathcal{M}}_{kji} = 0 \tag{4.23}$$

$$\hat{\mathcal{M}}_{kji} = \mathbf{S}_{3D} \tilde{\mathcal{M}}_{kji} \tag{4.24}$$

$$\mathbf{W}_k^T \mathbf{S}_{3D}^{-1} \hat{\mathcal{M}}_{kji} = \hat{\mathbf{W}}_k^T \hat{\mathcal{M}}_{kji} = 0$$

$$\mathbf{W}_k = \mathbf{S}_{3D}^T \hat{\mathbf{W}}_k \tag{4.25}$$

Normalising these sets of data to have centroids at zero and average value in each direction $\sqrt{2}$ improves stability [45]. Therefore, using this transformation for Equation 4.21, we perform the procedure of Section 4.4.1 by replacing $\tilde{\mathbf{m}}$ with $\hat{\mathbf{m}}$ and recover the actual values of $\mathbf{K}_0$ with Equation 4.22. Similarly for Equation 4.24, we perform the procedure of Section 4.4.1 replacing $\tilde{\mathcal{M}}$ with $\hat{\mathcal{M}}$, and recover $\mathbf{W}_k$ with Equation 4.25. We see in Section 4.6 that this second normalisation greatly improves results.

Global bundle adjustment

We further improve this with a final BA stage, which applies NLLS refinement to all intrinsic, extrinsic, and world point parameters, globally minimising reprojection error [154]. Although we already have two refinement stages, world points have only been refined with respect to the first camera pair and the camera parameters subsequent nodes have only been refined indirectly through the planes at infinity. Global BA directly imposes more geometric meaning to our refinement. The cost function for this stage is

$$\sum_{k=0}^{K} \sum_{i=1}^{N} \sum_{j=1}^{M} ||\mathbf{m}_{kji} - \pi_{kji}(\mathbf{K}_k, \xi_k \ \mathbf{M}_{1i}, \ \theta_i, \ \phi_i)||^2. \tag{4.26}$$

where π_{kji} reprojects the point $\mathbf{M}_{ji}$ onto the image of node k using Equations 4.1–4.2 and θ_i and ϕ_i from Equation 4.19. Although refining a large number of parameters, $(10K + 5M + 4)$, the prior normalisation techniques generally reduce global error such that this is relatively fast, as will be shown in Section 4.6.

4.5 Calibrating a distributed camera sensor network

For use in an ad-hoc network, we adapted our improved 1D calibration algorithm to perform distributed processing using a combination of ADMM and GaBP. In this section, we first introduce consensus with ADMM and GaBP, then present our work to adapt the problem to this framework.

4.5.1 General consensus alternating direction method of multipliers

Consider the objective of finding the vector of variables $\mathbf{z} \in \mathbb{R}^n$ that minimises a sum of objectives $f_1(\mathbf{x}_1) + ... + f_K(\mathbf{x}_K)$ separable in $\mathbf{x}_k$, where $\mathbf{x}_k \in \mathbb{R}^{n_k}$ corresponds to a subvector of $\mathbf{z}$ by some mapping of components $\mathcal{Q}(k,l)$. Using the notation $(\tilde{\mathbf{z}}_k)_l = \mathbf{z}_{\mathcal{Q}(k,l)}$, consensus between local variables and the global vector is achieved when $\mathbf{x}_k = \tilde{\mathbf{z}}_k$ [215]. That is,

$$\begin{aligned} &\underset{\mathbf{z},\mathbf{x}_1,\cdots,\mathbf{x}_K}{\text{minimise}} && \sum_{i=1}^{K} f_k(\mathbf{x}_k) \\ &\text{subject to} && \mathbf{x}_k - \tilde{\mathbf{z}}_k = 0, \quad k = 0, ..., K. \end{aligned} \tag{4.27}$$

For ADMM we express this problem with the augmented Lagrangian of Equation 4.28, with Lagrangian multiplier $\mathbf{y}_k \in \mathbb{R}^{n_k}$ and diagonal penalty matrix $\Lambda_k = diag(\lambda_{k1}, ..., \lambda_{kn_k})$ with all $\lambda_{kl} > 0$ and $\|\mathbf{r}\|^2_\Lambda = \mathbf{r}^T\Lambda\mathbf{r}$ [215],

$$\mathcal{L}_\Lambda(\mathbf{x},\mathbf{z},\mathbf{y}) = \sum_{k=0}^{K} f_k(\mathbf{x}_k) + \mathbf{y}_k^T(\mathbf{x}_k - \tilde{\mathbf{z}}_k) + \frac{1}{2}\|\mathbf{x}_k - \tilde{\mathbf{z}}_k\|^2_{\Lambda_k}. \tag{4.28}$$

ADMM solves Equation 4.27 by iterating over equations Equations 4.29–4.31.

$$\mathbf{x}_k^{(t+1)} := \underset{\mathbf{x}_k}{\text{argmin}}(f_k(\mathbf{x}_k) + \mathbf{y}_k^{(t)T}(\mathbf{x}_k - \tilde{\mathbf{z}}_k^{(t)}) + \frac{1}{2}\|\mathbf{x}_k - \tilde{\mathbf{z}}_k^{(t)}\|^2_{\Lambda_k}) \tag{4.29}$$

$$z_q^{(t+1)} := \frac{\sum_{\mathcal{Q}(k,l)=q}(\mathbf{y}_k^{(t)})_l + \lambda_{kl}(\mathbf{x}_k^{(t+1)})_l}{\sum_{\mathcal{Q}(k,l)=q}\lambda_{kl}} \tag{4.30}$$

$$\mathbf{y}_k^{(t+1)} := \mathbf{y}_k^{(t)} + \Lambda_k(\mathbf{x}_k^{(t+1)} - \tilde{\mathbf{z}}_k^{(t+1)}) \tag{4.31}$$

Considering the sum of related components of $\mathbf{y}$ we get,

$$\begin{aligned} \sum_{\mathcal{Q}(k,l)=q}(\mathbf{y}_k^{(t+1)})_l &:= \sum_{\mathcal{Q}(k,l)=q}\Big((\mathbf{y}_k^{(t)})_l + \lambda_{kl}((\mathbf{x}_k^{(t+1)})_l - z_q^{(t+1)})\Big) \\ &= \sum_{\mathcal{Q}(k,l)=q}\Big((\mathbf{y}_k^{(t)})_l + \lambda_{kl}(\mathbf{x}_k^{(t+1)})_l\Big) - z_q^{(t+1)}\sum_{\mathcal{Q}(k,l)=q}\lambda_{kl}. \end{aligned}$$

Substituting in the definition of z_q from Equation 4.30 we get $\sum_{\mathcal{Q}(k,l)=q}(\mathbf{y}_k)_l = 0$ after the first iteration. This reduces each z_q to a penalty-weighted average,

$$z_q^{(t+1)} := \frac{\sum_{\mathcal{Q}(k,l)=q} \lambda_{kl}(\mathbf{x}_k^{(t+1)})_l}{\sum_{\mathcal{Q}(k,l)=q} \lambda_{kl}}. \tag{4.32}$$

The joint global minimum of all local objective functions is found by iteratively solving Equations 4.29 and 4.31 using only local data and Equation 4.32 with a distributed average. We used a global and local component for $\lambda_{kl} = \lambda_{global}\lambda_{local,kl}$ where $\lambda_{local,kl}$ was inverse variance from our error model. Global components cancel out for Equation 4.32 giving a variance-weighted average.

4.5.2 Gaussian belief propagation

We can interpret the distributed variance-weighted average of Equation 4.32 as a probabilistic problem where all $(\mathbf{x}_k)_l$ for $\mathcal{Q}(k,l) = q$ are noisy measurements of true state z_q. As in Chapter 3, this can be solved by considering the joint density in Equation 4.33 with node potentials ϕ_i and edge potentials ψ_{ij}. Node potentials ϕ_k represent local evidence for $\mathbf{z}$, and edge potentials ψ_{ki} are discussed in Section 4.5.3. The pairwise joint density in Equation 4.33 ignores the effect of the first camera pair on subsequent estimations, but this effect is minor. Here i, j, and k are node indices.

This is solved using belief propagation by passing messages $m_{i\to j}^{(t)}(z_j)$, given in Equation 4.34, from node i to node j across edges $(i,j) \in \mathcal{E}$ at every iteration t. The belief at each iteration, given in Equation 4.35 converges towards a global estimate for z_i. Section 3.6 outlines the relationship between the problem formulation and belief propagation in greater detail.

$$p(\tilde{\mathbf{z}}_0, ..., \tilde{\mathbf{z}}_K) \propto \prod_{k\in\mathcal{V}} \phi_k(\tilde{\mathbf{z}}_k) \prod_{(k,i)\in\mathcal{E}} \psi_{ki}(\tilde{\mathbf{z}}_k, \tilde{\mathbf{z}}_i) \tag{4.33}$$

$$m_{k\to i}^{(t)}(\tilde{\mathbf{z}}_i) \propto \int_{\tilde{\mathbf{z}}_k} \phi_k(\tilde{\mathbf{z}}_k)\psi_{ki}(\tilde{\mathbf{z}}_k, \tilde{\mathbf{z}}_i) \prod_{j\in\mathcal{N}_k\setminus i} m_{j\to k}^{(t-1)}(\tilde{\mathbf{z}}_k)d\tilde{\mathbf{z}}_k \tag{4.34}$$

$$b_k^{(t)}(\tilde{\mathbf{z}}_k) \propto \phi_k(\tilde{\mathbf{z}}_k) \prod_{j\in\mathcal{N}_k} m_{j\to k}^{(t)}(\tilde{\mathbf{z}}_k) \tag{4.35}$$

Different to Chapter 3, this chapter uses the form of Gaussian belief propagation of Moallemi and Van Roy [201] with attenuation parameter β, solving Equation 4.32 using Gaussian densities for each z_q represented by inverse-variance P and mean μ,

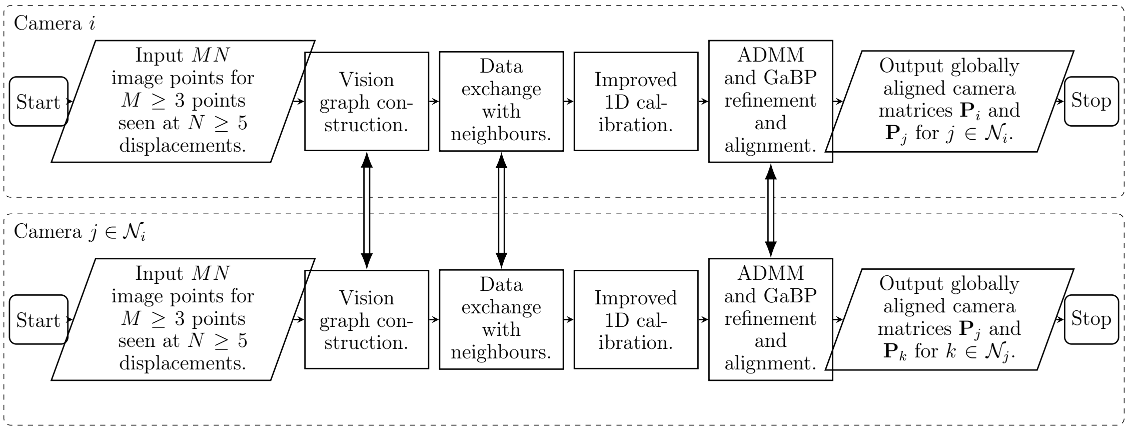

Camera i
Start
Input MN image points for $M \geq 3$ points seen at $N \geq 5$ displacements.
Vision graph construction.
Data exchange with neighbours.
Improved 1D calibration.
ADMM and GaBP refinement and alignment.
Output globally aligned camera matrices $\mathbf{P}_i$ and $\mathbf{P}_j$ for $j \in \mathcal{N}_i$.
Stop
Camera $j \in \mathcal{N}_i$
Start
Input MN image points for $M \geq 3$ points seen at $N \geq 5$ displacements.
Vision graph construction.
Data exchange with neighbours.
Improved 1D calibration.
ADMM and GaBP refinement and alignment.
Output globally aligned camera matrices $\mathbf{P}_j$ and $\mathbf{P}_k$ for $k \in \mathcal{N}_j$.
Stop

Figure 4.3: Full distributed algorithm, shown for two camera nodes. Communication between nodes at various stages is shown by double-lined arrows.

with messages given by Equations 4.36–4.37 and belief by Equation 4.38.

$$P^{(t)}_{k\to i} = (P_{kk} + \sum_{j\in\mathcal{N}_k\backslash i} P^{(t-1)}_{j\to k})/(1 + (P_{kk} + \sum_{j\in\mathcal{N}_k\backslash i} P^{(t-1)}_{j\to k})/\beta) \tag{4.36}$$

$$\mu^{(t)}_{k\to i} = (P_{kk}\mu_{kk} + \sum_{j\in\mathcal{N}_k\backslash i} P^{(t-1)}_{j\to k}\mu^{(t-1)}_{j\to k})/(P_{kk} + \sum_{j\in\mathcal{N}_k\backslash i} P^{(t-1)}_{j\to k}) \tag{4.37}$$

$$\mu^{(t)}_{k} = (P_{kk}\mu_{kk} + \sum_{j\in\mathcal{N}_k} P^{(t)}_{j\to k}\mu^{(t)}_{j\to k})/(P_{kk} + \sum_{j\in\mathcal{N}_k} P^{(t)}_{j\to k}) \tag{4.38}$$

We do this for each component of $\mathbf{z}$ taking μ_{kk} from $(\mathbf{x}_k)_l$ with P_{kk} its inverse variance. The converged μ^t_k is the consensus value of z_q used in the next iteration of ADMM. Standard GaBP only converges to an approximate mean for loopy graphs, and the more common method for computing Equation 4.30 is *average consensus* which does produce the correct mean [35]. However, due to the attenuation parameter this form of GaBP does produce the actual mean within a desired tolerance at a faster rate than average consensus [201]. It is for this reason that we compute Equation 4.32 in this manner.

4.5.3 Adapting 1D calibration for distributed processing

Using these concepts, we adapted the calibration to a distributed network. This involved building a vision graph, performing local calibrations at each node, then refining and aligning the local estimates using ADMM and GaBP with a robust error model. The full process is shown in Figure 4.3.

Vision graph for cluster-based calibration

For the vision graph, assuming good temporal synchronisation, we assign an edge between cameras when they share a sufficient number of observations. The minimum number is 5, however a higher threshold improves accuracy. Nodes exchange observations with neighbours to perform the 1D calibration locally, excluding final BA. Each node considers itself as 'Camera 0' and computes the extrinsic parameters for its neighbours relative to itself.

Local and global objective functions

The global BA objective is given in Equation 4.26. The corresponding local BA actually performed at each node is the same but on a subset of cameras and within the node's local frame. We use indicator variable w_{kli} equal to one if neighbouring cameras k and l both observed calibration object displacement i and zero otherwise, and we use superscript k to denote a local estimation at node k, initially differing from corresponding values at different nodes.

$$f_k(\mathbf{x}_k) = \sum_{l \in \mathcal{N}_k \cup k} \sum_{i=1}^{N} \sum_{j=1}^{M} w_{kli} || \mathbf{m}_{lji} - \pi_{lji}(\mathbf{K}_l^k, \xi_l^k \ \mathbf{M}_{1i}^k, \ \theta_i^k, \ \phi_i^k) ||_2^2 \tag{4.39}$$

The global objective is the sum of the local objectives, which is to substitute $f_k(\mathbf{x}_k)$ into Equation 4.27. Here, $\mathbf{x}_k$ is a vector containing all parameters being estimated locally. Assuming that the neighbours of node k are $k+1, \cdots, k+d_k$ to simplify notation and using $\mathbf{a}_l^k = [\alpha_l^k, \beta_l^k, u_l^k, v_l^k]$, we have

$$\mathbf{x}_k = [\mathbf{a}_k^k, \cdots, \mathbf{a}_{k+d_k}^k, \xi_{k+1}^k, \cdots, \xi_{k+d_k}^k, \mathbf{M}_{11}^k, \theta_1^k, \phi_1^k, \cdots, \mathbf{M}_{1N}^k, \theta_N^k, \phi_N^k]^T. \tag{4.40}$$

Compared to Equation 4.26, the data from each node k is now being repeated in each of its neighbours, so to keep these two objectives the same we down-weight residuals by setting all non-zero w_{kli} to $(d_l + 1)^{-1}$.

Frame alignment and error propagation

There are two remaining issues with the distributed refinement and alignment. Firstly, nodes have measurements of world points and neighbour's poses in their local frame. To align these estimates we cannot just average them, but must use a node's current beliefs on its neighbour's poses in the edge potentials when receiving messages. Therefore, nodes send these messages in their local frame but the target nodes transform them into their own frame. Consider a message sent from node k to i about the pose of node j, which corresponds to ξ_j^k and has the basis frame of k denoted by $\mathfrak{B}_k$. Node i wants to compare this to its local estimate ξ_j^i in basis $\mathfrak{B}_i$ and must use its current belief about the pose of k corresponding to ξ_k^i. That is,

$$m_{k\to i}^{(t)}(\xi_j)_{\mathfrak{B}_i} = m_{k\to i}^{(t)}(\xi_j)_{\mathfrak{B}_k} \circ b_i^{(t)}(\xi_k). \tag{4.41}$$

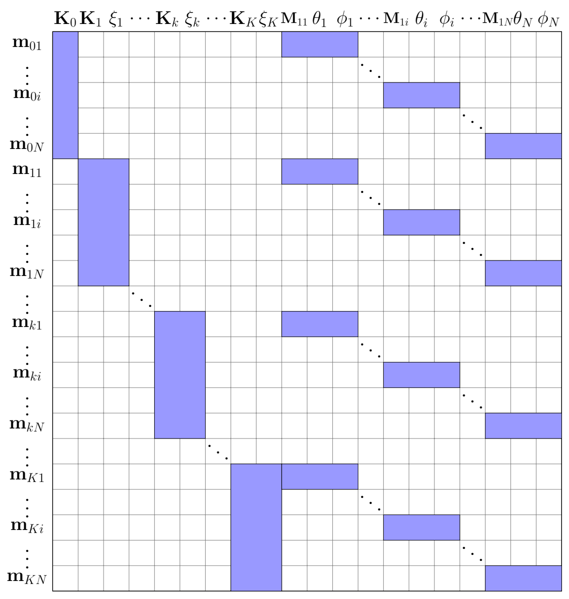

$\mathbf{K}_0$ $\mathbf{K}_1$ ξ_1 $\cdots$ $\mathbf{K}_k$ ξ_k $\cdots$ $\mathbf{K}_K$ ξ_K $\mathbf{M}_{11}$ θ_1 ϕ_1 $\cdots$ $\mathbf{M}_{1i}$ θ_i ϕ_i $\cdots$ $\mathbf{M}_{1N}$ θ_N ϕ_N
$\mathbf{m}_{01}$
$\mathbf{m}_{0i}$
$\mathbf{m}_{0N}$
$\mathbf{m}_{11}$
$\mathbf{m}_{1i}$
$\mathbf{m}_{1N}$
$\mathbf{m}_{k1}$
$\mathbf{m}_{ki}$
$\mathbf{m}_{kN}$
$\mathbf{m}_{K1}$
$\mathbf{m}_{Ki}$
$\mathbf{m}_{KN}$

Figure 4.4: Jacobian sparsity of bundle adjustment, with $\mathbf{m}_{ki} = [\mathbf{m}_{k1i}^T \cdots \mathbf{m}_{kMi}^T]^T$ for brevity.

where $\circ$ denotes composition, $\xi_j^k \circ \xi_k^i = log_{SE(3)}(exp_{\mathfrak{se}(3)}(\xi_j^k) exp_{\mathfrak{se}(3)}(\xi_k^i))$.

Secondly, as this is a non-convex problem, ADMM is not guaranteed to converge to the global minimum, but instead a local minimum or not converge at all. We can improve convergence by incorporating a robust error model for the variance-weighted average for Equation 4.32. We assumed zero-mean additive Gaussian noise with variance σ^2 for our image points and transformed this into parameter error through back-propagation then into frame-aligned messages with forward-propagation. Nodes have a $2NM(d_k+1) \times 2NM(d_k+1)$ diagonal covariance matrix $\Sigma_\mathbf{m} = diag(\sigma^2)$ for image points $\mathbf{m}$. To get parameter covariance $\mathbf{x}_k$, $\Sigma_{\mathbf{x}_k}$ we back-propagate through the reprojection function π_{lji} in Equation 4.39 whose Jacobian $\mathbf{J}$ has the sparsity pattern shown in Figure 4.4. This back-propagation has the form of Equation 4.42 where $(.)^+$ is the Moore–Penrose pseudo-inverse. For each variable in $\mathbf{x}_k$ its inverse-variances P_{kk} in Equations 4.36–4.38 is the associated diagonal value of $\Sigma_{\mathbf{x}_k}$.

$$\Sigma_{\mathbf{x}_k} = (\mathbf{J}^T \Sigma_\mathbf{m}^{-1} \mathbf{J})^+ \tag{4.42}$$

We also consider error propagation for the alignment of messages in Equation 4.41, where we need to forward-propagate the inverse-variance P. Taking a second order approximation, we can propagate the covariance of an $\mathfrak{se}(3)$ pose by its adjoint [256], giving Equations 4.43–4.44. We do an equivalent transform for world points, where the adjoint is simply $\mathbf{R}$.

$$\mu^{(t)}_{k\to i}(\xi_j)_{\mathfrak{B}_i} = \mu^{(t)}_{k\to i}(\xi_j)_{\mathfrak{B}_k} \circ \mu^{(t)}_i(\xi_k) \tag{4.43}$$

$$P^{(t)}_{k\to i}(\xi_j)_{\mathfrak{B}_i} = [\mathrm{Adj}(\xi_k) P^{(t)}_{k\to i}(\xi_j)^{-1}_{\mathfrak{B}_k} \mathrm{Adj}(\xi_k)^T + P^{(t)}_i(\xi_k)^{-1}]^{-1} \tag{4.44}$$

$$\text{where} \quad \mathrm{Adj}(\xi) = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}$$

Full refinement and alignment process

The full process of refining and aligning the local calibration estimates iterates as follows until convergence:

1. Each node k finds $\mathbf{x}_k^{(t+1)}$ from Equation 4.29 with $f_k(\cdot)$ from Equation 4.39. As with [223], we only do one of this minimisation iteration per outer iteration.
2. $\tilde{\mathbf{z}}_k^{(t+1)}$ is found for Equation 4.32 using the GaBP procedure.
   (a) If any GaBP message corresponds to a pose or world point, it is transformed between communicating frames by Equations 4.43–4.44.
3. Each node updates its Lagrangian multiplier $\mathbf{y}_k^{(t+1)}$ by Equation 4.31.

Nodes initialise $\tilde{\mathbf{z}}_k^{(0)} = \mathbf{x}_k^{(0)}$ and $\mathbf{y}_k^{(0)} = \mathbf{0}$. As mentioned, the local component of the penalties is determined from the inverse variance $\lambda_{local,kl} = (\Sigma_{\mathbf{x}_k})_l$. Additionally, we found good performance using the scheme from [223] where λ_{global} is updated each iteration as $\lambda^{(t+1)}_{global} = (1+\eta)\lambda^{(t)}_{global}$ for $\lambda^{(0)}_{global} = 10^{-3}$ and $\eta = 0.01$. We also found fast convergence with $\beta = 10^4$.

4.6 Experimental results

We ran simulations to test the effects of noise, pattern displacements, number of calibration points, compared centralised and distributed algorithms, and explored the effect of message loss. Then, we verified these results on real images.

4.6.1 Simulation data

Simulations had ten cameras with $\alpha = \beta = 655.0$px and principle point at the centre of the 1024×768px image. Each camera faced outwards evenly across an arc for $z > 0$ of a circle of 2.7m radius centred at $(0, 0, -2.5)$m, further translated in each axis by $[-0.1, 0.1]$m and perturbed about their y-axis by $[-\pi/18, \pi/18]$ radians with uniform sampling. A 20cm calibration object had $\mathbf{M}_{1i} = [X_{1i}, Y_{1i}, Z_{1i}]^T$ sampled uniformly from a concentric arc of radius of 3.5m, with rotations $\theta_i \in [\pi/6, 5\pi/6]$ and $\phi_i \in [\pi, 2\pi]$ radians. An example layout is given in Figure 4.5. Error in intrinsic

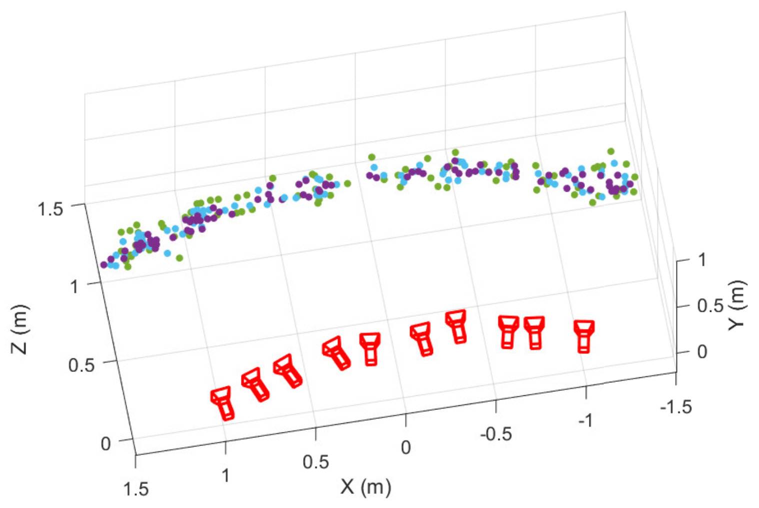
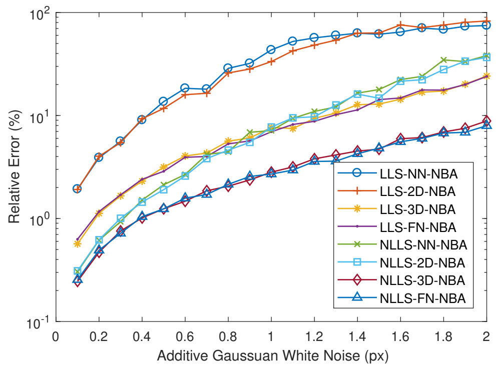


Figure 4.5: The ground truth layout for one trial of our simulation, using ten cameras.

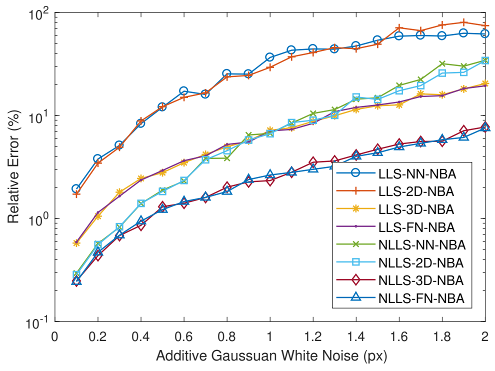
parameters was measured relative to the correct α, and errors in extrinsic parameters were relative to their true values. We used an equidistant three-point calibration object in all tests but the third, and 30 pattern displacements in all but the second. All simulations were run in MATLAB for Windows using a 3.4GHz Intel i7-6700 CPU.

Firstly observing noise, we tested the algorithm without normalisation (`NN`), 2D normalisation only (`2D`), 3D normalisation only (`3D`), and full normalisation (`FN`). These were repeated with (`NLLS`) and without (`LLS`) NLLS of Equations 4.17–4.18, then with (`BA`) and without (`NBA`) the final BA, producing sixteen configurations each tested for 1000 trials on synthetic data corrupted by 0.1 to 2.0px of Gaussian noise. Results are shown in Figure 4.6 and 4.7, and average processing time in Figure 4.8.

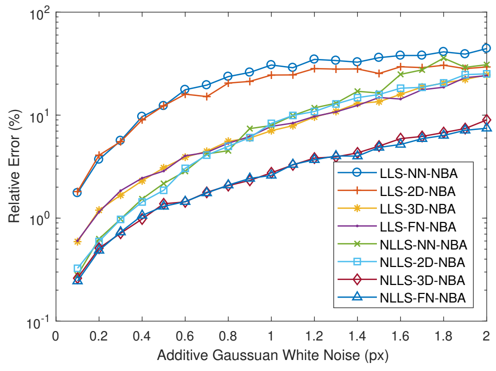
The results shown in Figure 4.6 without the final BA stage demonstrate the effects of the two types of normalisation as well as the NLLS step. The two worst performing variants were `LLS-NN-NBA` and `LLS-2D-NBA`, with their similar performance demonstrating that the 2D normalisation had little effect. Comparatively, the `LLS-3D-NBA` and `LLS-FN-NBA` resulted in more accurate results across all noise levels. Again, their similar results show that the 2D normalisation was having little effect. With the introduction of the NLLS step, `NLLS-NN-NBA` and`NLLS-2D-NBA` had better performance than the LLS variants of `LLS-3D-NBA` and `LLS-FN-NBA` at low levels of noise but not at high levels of noise. This shows that the 3D normalisation step is more effective across a range of noise levels whether with or without the NLLS step, however, the quality of the NLLS refinement degrades with noise if the 3D normalisation is excluded. This has implications in robust long-term opera-



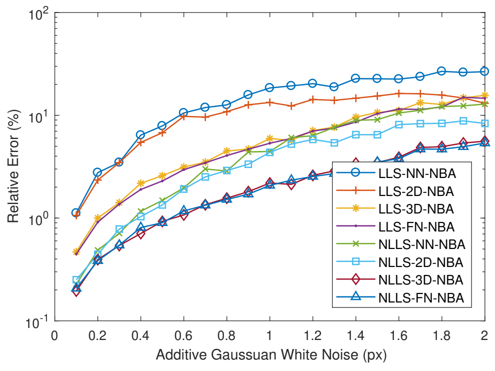
(a) Relative error in α



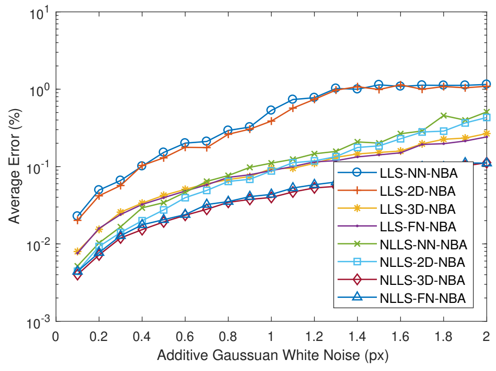
(b) Relative error in β



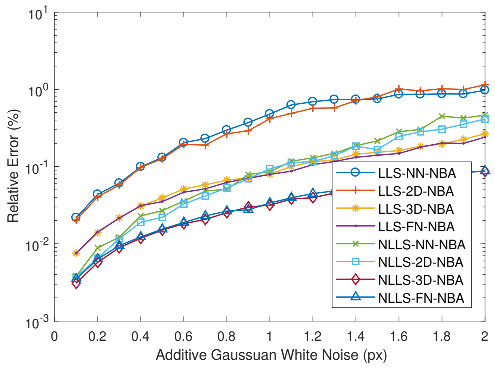
(c) Relative error in c_u



(d) Relative error in c_v

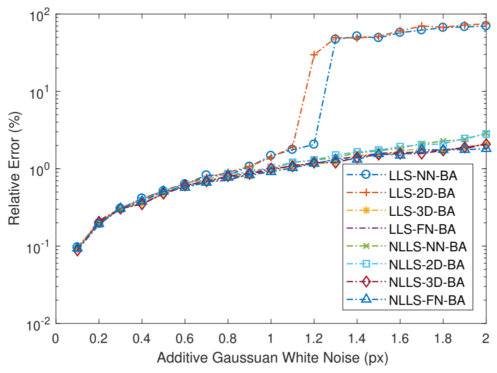


(e) Error in translation

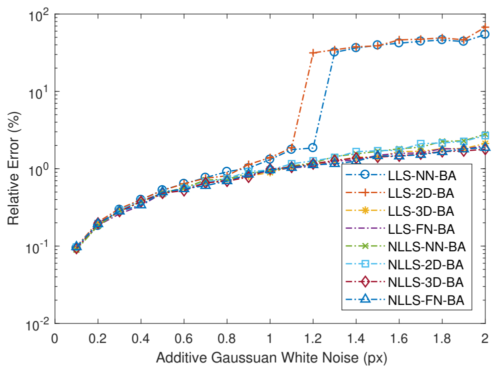


(f) Error in rotation

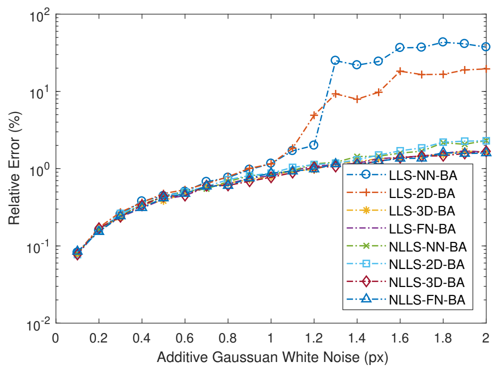
Figure 4.6: Average error in 1000 trials for algorithms without final bundle adjustment.



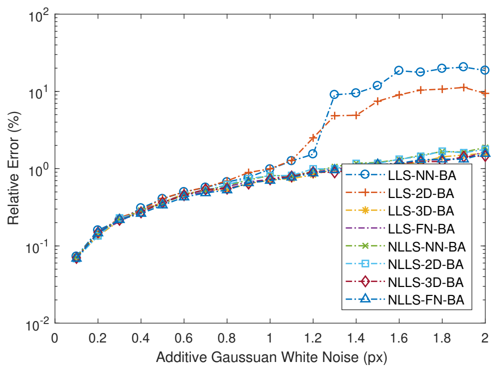
(a) Relative error in α



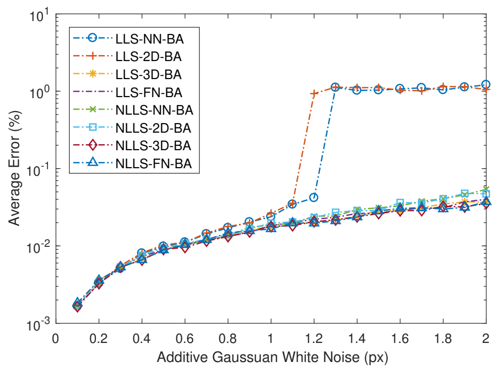
(b) Relative error in β



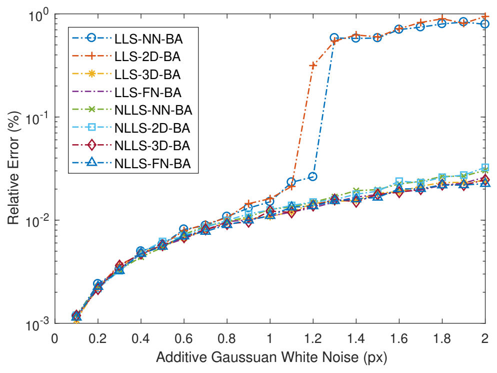
(c) Relative error in c_u



(d) Relative error in c_v



(e) Error in translation



(f) Error in rotation

Figure 4.7: Average error in 1000 trials for algorithms with final bundle adjustment.

tion of the algorithm on real-world camera systems where accumulation of dirt on the lens would result in noisy images, showing that the 3D normalisation would be crucial in such a use-case. These results show that the NLLS of Equations 4.17 and 4.18 as well as the 3D normalisation of Equation 4.24 greatly improved accuracy, whereas the effect of the 2D normalisation of Equation 4.21 was minor, suggesting that the plane at infinity estimation is quite susceptible to noise.

BA had the greatest effect, bringing all tests except `LLS-NN-BA` and `LLS-2D-BA` to a similar results. These two LLS variants have a clear failure point after BA at 1.2px of noise suggesting that without at least one of either the 3D normalisation or the NLLS refinement then the original estimate prior to BA has too much error and is susceptible to falling into a local minimum. This demonstrates that for robust long-term operation where accumulation of dirt would result in higher levels of effective noise that 3D normalisation or NLLS refinement is needed in combination with BA.

The execution times shown in Figure 4.8 show the effects of the two types of refinement when combined with the two types of normalisation. When using no normalisation or 2D normalisation with only one of either NLLS or BA the execution time grows fairly quickly with noise level due the greater number of iterations required before the LM stopping condition is met. Applying only the BA to the LLS solution with 3D normalisation present in either `LLS-3D-BA` or `LLS-FN-BA` also has the execution time grow noticeably with error level, but at a much slower rate. This shows that the 3D normalisation cuts down the number of iterations required by BA substantially. With the combination of NLLS refinement with either 3D normalisation present or the final BA then the NLLS stage remains fairly constant in execution time across all noise levels. Overall, for noise below 1.5px, it was faster to apply our improvements but skip Equation 4.17 and Equation 4.18 without much loss in accuracy.

Therefore, the results both in noise and execution time show that combining 3D normalisation with the final BA but without the NLLS step is the best choice for application on lower-powered hardware for long-term use.

The second simulation differed the number of pattern displacements, varying from 10 to 100 with a constant 1.0px of Gaussian noise. The errors and processing times for 1000 trials are given in Figure 4.9a. The third simulation varied the number of points on the calibration object between 3 and 7 at 1.0px of Gaussian noise, with results reported in Figure 4.9b. Both these tests saw a linear increase in processing time, with increasing pattern displacements having a greater effect on accuracy compared to more calibration points, however this sees diminishing returns.

Next, six alternatives were compared to evaluate the distributed algorithm — the improved centralised (`LLS-FN-BA`) and equivalent distributed (`LLS-FN-BA-DIST`) algorithms, the original centralised (`NLLS-NN-NBA`) [87] and equivalent dis-

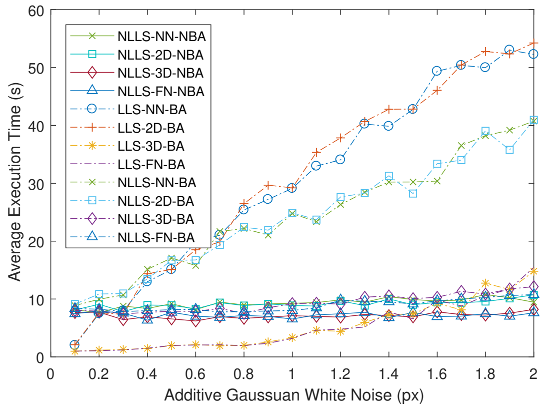
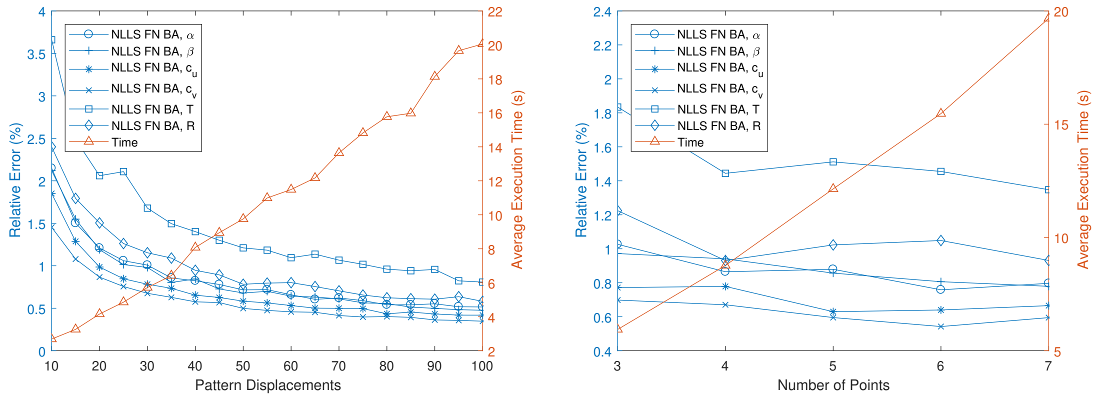


Figure 4.8: Average time for each algorithm to complete at different noise levels.



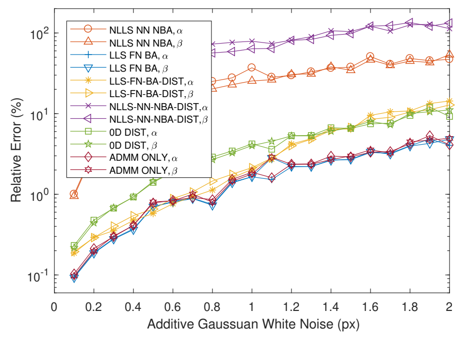
(a) Relative error and average execution times for different numbers of displacements

(b) Relative error and average execution times for different numbers of points

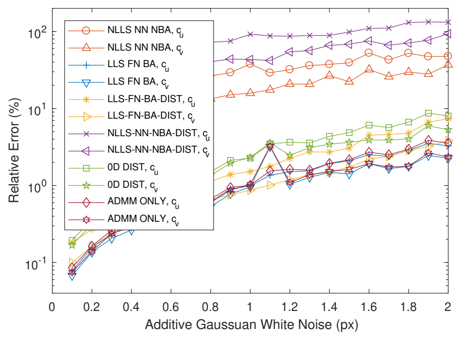
Figure 4.9: Average errors and processing times for 1000 trials in second and third tests.

tributed (`NLLS-NN-NBA-DIST`), as well as the distributed self-calibration of Devarajan et al. (`OD-DIST`) [7]. We also considered a centralised algorithm with only the global BA distributed with ADMM across the outer summation, not requiring frame alignment (`ADMM-ONLY`). Results are presented in Figure 4.10, showing that our centralised and distributed algorithms are similarly accurate, both much better than the original algorithm and slightly better than the self-calibration. Also the distributed per-processor execution time is much lower. The unimproved distributed algorithm was less accurate than its centralised counterpart and at higher noise levels was even slower per node due to being caught in NLLS, showing the need for normalisation. Finally, the ADMM centralised solution performed roughly the same as our centralised algorithm. This was not truly distributed since all but the BA was computed centrally, but shows the effects of ADMM and pairwise GaBP on the accuracy.

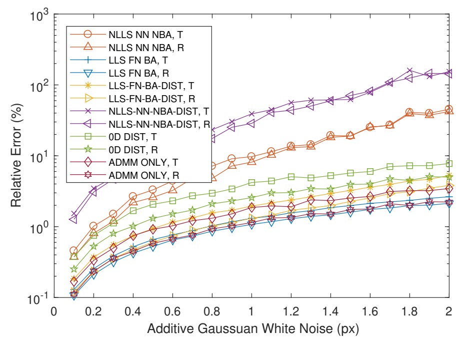
Finally, the effect of message-loss was tested on the best performing distributed



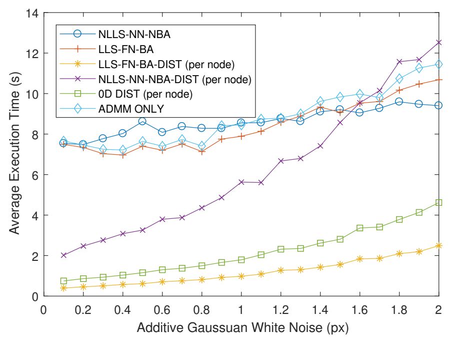
(a) Relative error for focal length parameters



(b) Relative error for principal point parameters



(c) Relative error for extrinsic parameters



(d) Average execution time for each algorithm

Figure 4.10: Average errors and executions times for 1000 trials in fourth simulation — comparing original algorithm, improved algorithm and distributed implementation.

algorithm (LLS-FN-BA-DIST). Although this algorithm operates on the assumption that messages for the distributed alignment are effectively routed to the correct node, in real-world situations it is possible that a message might not arrive due to time delays and a range of environmental factors. Therefore, the relative error was measured at a constant 1.0px of Gaussian noise with the percentage of messages lost varying from 0% to 50%, with results shown in Figure 4.11. As can be seen, the error in each parameter increases steadily as the message loss increases, however, the algorithm is fairly tolerant of low levels of message loss with minimal effects below 15%.

4.6.2 Real data

Finally, we verified our algorithm on real data using ten Raspberry Pi 3s and camera modules. 'Ideal' intrinsic parameters from the data-sheets are given in Table 4.1. Although not precisely ground truth values, modern cameras are of high enough quality that this is a decent baseline for comparison. Due to this lack of a ground truth for intrinsics, we also compared root-mean-squared reprojection error of the

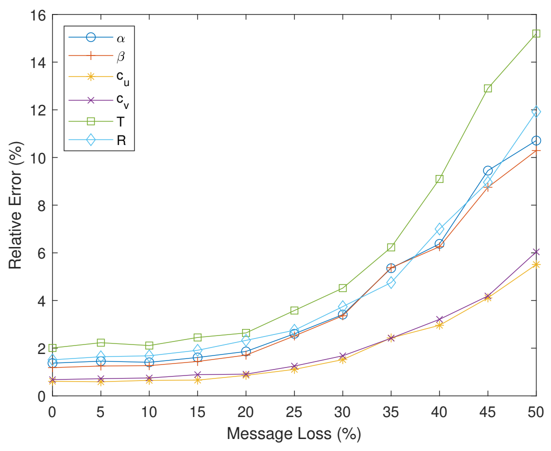


Figure 4.11: Relative error for the distributed algorithm at 1.0px of Gaussian noise and differing levels of message loss.

Table 4.1: Ideal intrinsics from camera data-sheets (in pixels).

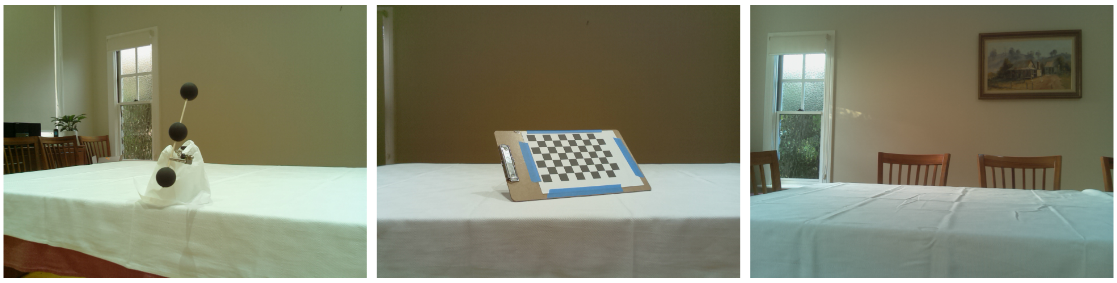
Camera	α	β	c_u	c_v
Cameras 0-4	634	634	320	240
Cameras 5-7	612	612	320	240
Cameras 8-9	501	501	320	240

reconstructed world points. The cameras were in an arc, with centre distances and rotations relative to camera 0 reported in Table 4.2. The calibration object was 205mm long, with three equidistant 40mm balls used as points. For the distributed algorithm the vision graph was determined by common visibility of the calibration object at 90 pattern displacements, a minimum of 30 visible at each camera. We compared the original algorithm [87], and our centralised and distributed algorithms. We also compared this to 2D calibration with NLLS [13], and distributed self-calibration [7]. Example images are shown in Figure 4.12.

A summary of results is given in Table 4.3, with full results in Table 4.5. We used the same error metrics as the simulations. As the self-calibration cannot determine scene scale, it's average translation was scaled to the average ground truth for comparison. Our improved algorithm estimates the extrinsic parameters noticeably better than 2D calibration in many parameters and has much higher accuracy than

Table 4.2: Ground truth for centre distances and y-axis rotations, relative to Camera 0.

Camera	1	2	3	4	5	6	7	8	9
Distance (mm) ±0.5mm	134	263	399	519	638	752	895	1064	1228
Rotation (deg) ±0.5°	9	14	20	26	31	40	48	48	62



(a) 1D calibration object **(b)** 2D calibration object **(c)** 0D calibration scene

Figure 4.12: Example images from the experimental test showing the three types of calibration object (or lack thereof).

Table 4.3: Relative difference to ground truth and ideal results averaged per algorithm, and RMS reprojection error.

Algorithm	α	β	c_u	c_v	T	R_y	RMSE
2D [13]	0.63%	0.75%	**1.15%**	**1.02%**	2.65%	2.90%	**0.126px**
1D de França [87]	2.21%	**0.45%**	1.76%	3.28%	1.15%	2.49%	0.468px
0D Distributed [7]	0.90%	0.88%	1.65%	2.37%	1.38%	2.46%	0.223px
1D Improved	**0.46%**	0.60%	1.47%	2.30%	**0.46%**	2.01%	0.187px
1D Distributed	0.59%	0.68%	1.35%	2.60%	0.56%	**2.00%**	0.171px

the original 1D algorithm. We can also see that the centralised and distributed algorithms again performed comparably.

The execution time analysis seen in Figure 4.10d was also repeated for the real data, shown in Table 4.4. As can be seen, the per-processor execution time is much lower for the distributed algorithm, allowing the processing to be effectively spread over a number of low-powered processors.

4.7 Summary

We have proposed a 1D calibration and localisation algorithm well-suited for low-cost distributed ad-hoc camera networks. This is presented as an improved version

Table 4.4: Average execution time for the different configurations of the calibration algorithm when applied to real data.

Algorithm	Time (s)
`NLLS-NN-NBA`	8.31
`LLS-FN-BA`	7.49
`ADMM-ONLY`	7.47
`NLLS-NN-NBA-DIST` (per node)	2.92
`LLS-FN-BA-DIST` (per node)	0.87
`0D-DIST` [7] (per node)	1.17

Table 4.5: Experimental measurements for each camera across the four algorithms

Algorithm	Camera	α	β	c_u	c_v	T (mm)	R_y (deg)
2D [13]	0	636.63	639.51	324.63	236.25		
	1	642.54	645.40	322.67	217.29	140.01	8.51
	2	631.15	632.36	316.80	244.20	270.21	14.17
	3	633.86	636.21	325.26	240.43	407.95	19.69
	4	635.23	637.52	317.66	231.35	532.15	25.94
	5	609.56	611.83	317.96	239.26	652.40	29.02
	6	608.37	610.58	297.71	249.33	767.39	39.00
	7	607.71	609.90	328.45	247.75	914.94	46.69
	8	504.06	506.51	312.57	235.20	1091.23	48.99
	9	508.76	510.71	310.71	239.62	1262.31	59.43
1D de França [87]	0	643.58	632.58	311.76	258.55		
	1	650.44	637.25	314.320	250.03	133.98	8.38
	2	648.69	634.65	316.11	265.23	259.69	14.32
	3	652.60	637.51	322.26	260.19	393.81	19.60
	4	656.64	641.55	319.45	248.14	514.70	25.93
	5	626.07	609.55	333.07	272.05	630.44	30.32
	6	623.09	609.14	327.01	275.94	740.91	41.39
	7	618.05	606.45	360.86	264.67	881.84	49.04
	8	513.21	501.73	324.32	253.22	1050.17	49.38
	9	509.33	501.13	337.48	250.29	1208.76	61.89
0D Distributed [7]	0	623.43	623.97	323.17	258.66		
	1	627.64	634.67	327.42	241.41	137.12	8.35
	2	639.80	624.25	325.22	264.75	265.34	14.51
	3	630.11	631.94	330.46	260.18	402.43	19.98
	4	633.73	627.13	322.13	246.36	527.85	26.29
	5	611.56	613.34	338.25	262.10	642.41	29.49
	6	606.29	608.58	316.06	261.17	766.54	40.71
	7	604.02	608.80	347.73	259.05	899.08	47.60
	8	493.53	492.49	311.27	247.60	1086.95	48.46
	9	505.77	494.97	330.69	243.91	1244.85	60.97
1D Improved	0	631.91	629.76	323.89	256.64		
	1	634.56	633.11	325.74	241.28	135.74	8.46
	2	630.63	629.26	325.53	263.59	263.27	14.36
	3	633.09	631.67	328.75	256.97	399.84	19.52
	4	635.85	635.14	323.62	244.86	522.99	25.82
	5	608.08	606.11	335.34	263.97	641.12	30.16
	6	606.70	606.09	318.61	264.39	753.70	40.57
	7	603.51	603.43	349.49	258.50	897.65	48.13
	8	500.22	498.30	314.69	247.15	1068.53	48.47
	9	501.70	500.90	328.51	243.12	1232.11	61.30
1D Distributed	0	630.39	629.10	323.56	258.23		
	1	635.27	635.31	326.86	240.81	134.66	8.59
	2	631.20	629.74	327.09	262.80	262.36	14.41
	3	628.96	628.11	326.55	259.62	399.82	19.71
	4	632.08	637.05	323.28	247.86	521.24	25.99
	5	603.57	607.26	336.87	265.26	643.72	30.28
	6	606.87	607.38	318.28	268.07	758.54	41.07
	7	604.97	604.14	347.03	260.94	900.30	48.37
	8	501.61	498.39	315.35	249.20	1074.02	49.28
	9	501.63	499.06	323.94	245.42	1232.24	61.62

of the algorithm proposed in Chapter 3, addressing the main issues of the fixed-point constraint on the motion of the calibration object and the ambiguous global objective of the bundle adjustment. In addressing these issues, an improved initialisation stage for the pipeline being discussed in this dissertation has been proposed.

Whilst there are many popular calibration methods, namely 2D and self-calibration, ours has a number of advantages. The lack of self-occlusion of the calibration object suits larger CSNs, the general motion makes it simple to use, and it can determine scene scale. Our contributions were to improve an existing 1D algorithm with better numerical conditioning and global BA. We do not consider radial distortion, but this can be done with the process of [13]. We also adapted the algorithm to a distributed network with global consensus using ADMM and GaBP with frame alignment. The algorithm requires at least two cameras taking five images each of a calibration object with at least three collinear points. We found minimal improvement beyond three points, and 30-40 images per camera provided good results.

Simulations showed that 3D normalisation greatly improved accuracy due to reducing the transferred effect of error in fundamental matrices. Global BA also significantly improved accuracy, removing the need for the other NLLS. Experimental results showed that the centralised and distributed variants of our algorithm were more accurate than 2D calibration for four of six parameters tested, and five out of six compared to the original 1D algorithm. We also saw that normalisation was important for convergence of the distributed solution. This distributed solution has benefits of resistance to node failure and scalability. Our algorithm provides highly accurate results, is simple to use, and is well suited to CSNs. In the future we plan to adapt the distributed localisation component of this algorithm to further problems such as visual odometry and SLAM, and further explore distributed approaches to bundle adjustment.

From this, it can be seen that an improved initialisation stage for the distributed robotic vision pipeline has been proposed.

Chapter 5

Direct visual odometry with binary descriptors[a]

Contents

5.1 Summary of contributions

- Considering the Census transform and Rank transform, a Lucas-Kanade-based visual odometry algorithm is developed that determines gradient and descent direction using Hamming weights of descriptors.
- The algorithm is extended to BRIEF, ORB, and BRISK descriptors, treating the entire robust descriptor as a single channel.
- An alternate form is explored that splits the descriptor into multiple channels.
- Evaluation is done on real data, with consideration of accuracy and computational time with sparse point selection.

[a]Chapter adapted from "B. Halloran *et al.*, 'Single and multi-channel direct visual odometry with binary descriptors,' in *Proceedings of ICIC 2019: Intelligent Computing Methodologies*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2019, pp. 86–98, ISBN: 978-3-030-26766-7. DOI: 10.1007/978-3-030-26766-7_9"

- All algorithms perform better than raw intensity, however, the alternative form did not perform as well as the single-channel form.
- ORB descriptors have the best accuracy, while BRIEF descriptors had the least computation time.

5.2 Introduction

Continuing on from the initialisation-stage algorithms presented in Chapters 3 and 4, regarding the three-stage pipeline presented in Figures 1.2 and 2.28, this chapter explores the single-robot visual odometry procedure utilised in the second stage. The primary goals for this is to incorporate the robust feature descriptors required in the global mapping of the third stage into a direct visual odometry algorithm for this second stage.

Visual odometry (VO) is one of the most active areas of research in computer vision. It is the process of estimating the relative poses of subsequent frames from a camera and forms the core of visual *simultaneous localisation and mapping* (V-SLAM) seen in algorithms such as LSD-SLAM [92], ORB-SLAM [100], SVO [102] and DSO [93].

There are two main types of VO — indirect and direct methods. Indirect methods use robustly matched feature correspondences to estimate poses [257], such as in MonoSLAM [88] and ORB-SLAM [100]. Direct methods perform whole-image alignment on intensity to estimate poses, generally using the Lucas-Kanade (LK) algorithm [147] and recently the more efficient Inverse Compositional (IC-LK) formulation [106]. Such direct VO algorithms include DTAM [89], LSD-SLAM [92] and DSO [93]. A middle ground between these methods is to use densely evaluated features in a direct VO framework. Standard direct methods rely on the *brightness consistency assumption* (BCA) which is not robust to illumination changes seen in real image sequences. Feature-based direct methods alleviate this using a *descriptor consistency assumption* (DCA) instead [258].

Our work explores direct tracking with robust binary descriptors such as BRIEF [152], ORB [101], and BRISK [151] which are efficient to compute and match. Many existing feature-based direct methods either use expensive features such as SIFT [259], or simpler binary descriptors such as the Census transform [260]. Our work involves a simple single-channel method for direct tracking with IC-LK and binary descriptors which performs extremely well on much larger descriptors and could be extended to multi-view systems.

5.2.1 Related work

There has been extensive work on improving the robustness of the LK algorithm, particularly for use in VO and SLAM. Some algorithms still operate directly on intensity values but also model changes in brightness, such as the pixel-wise lighting and shadow model of Silveira and Malis [261], and the affine lighting model of Engel et al. [93].

Feature-based LK has been used in areas including VO and optical flow. The SIFT Flow algorithm uses dense SIFT descriptors with LK on each channel [259]. Bristow and Lucey showed that while dense descriptors are poor predictors of the error surface they still have very good performance for gradient based methods [262]. Sevilla-Lara and Learned-Miller proposed distribution fields which 'explode' a single-channel image into many channels based on the intensity values, thus preventing smoothing from erasing small details [263]. They describe this form of DCA as channel consistency [264]. Crivellaro and Lepetit use a similar idea where they separate the image into channels for first and second order gradients which are further separated by sign [179]. Binary features, mainly the Census transform, have also seen use in LK. Although binary features are non-convex and non-differentiable, Alismail et al. showed that they can be used in LK by splitting up each bit of the descriptor into a different channel [265]. Their method had good performance particularly in low-light scenes but does not scale well to larger descriptors which could require as many as 512 channels. Recently, Park et al. showed that these Census-based methods were more effective than other descriptor-based methods on real-world images [266].

5.2.2 Contributions

We present a method for performing LK-based VO with binary features which operates only on a single channel, allowing the use of robust descriptors which are suitable for wide baseline matching and challenging lighting conditions. The primary goal of this chapter is to produce a robust single-robot visual odometry method for the second stage of the pipeline presented in Figures 1.2 and 2.28. Our contributions are:

- Considering the Census transform and its related Rank transform, we present an LK-based algorithm that compares the binary features using their Hamming distance but determines gradient and descent direction by comparing Hamming weight.
- We show that this algorithm achieves excellent performance when applied to BRIEF [152], ORB [101], and BRISK [151] descriptors, still operating on only

a single channel.

- We demonstrate an alternative multi-channel channel form, with still fewer channels than other methods, by using the Hamming weight of segments of the descriptor.

- Evaluations on real data demonstrate that our proposed method performs better than raw intensity and the Bitplanes method of Alismail et al. [265]. Furthermore, incorporating sparsely evaluated robust descriptors does not impact computational efficiency too much.

The remainder of the chapter is organised as follows: Section 5.3 covers the background information on the LK algorithm; Section 5.4 discusses our methods for tracking on binary descriptors; Section 5.5 presents our test results and analyses the performance; finally, Section 5.6 concludes the chapter and presents our direction for future work.

5.3 Preliminaries

This section outlines the preliminary concepts for solving the direct visual odometry problem, introduced in Section 2.4.1, using the Lucas-Kanade algorithm and its variants.

5.3.1 Notation

In this work we represent vectors as bold lower-case letters ($\mathbf{m}$), matrices as bold upper-case letters ($\mathbf{T}$). Functions, including images, are given by light upper-case letters (I). Using the set of valid pixels in an image as $\Omega \subset \mathbb{R}^2$ then we have images $I : \Omega \to \mathbb{R}^+$, depth maps $D : \Omega \to \mathbb{R}^+$, and depth variance maps $V : \Omega \to \mathbb{R}^+$ which all map pixels to intensity, inverse depth, and inverse depth variance respectively. Note that we use *inverse* depth d as opposed to depth z to parameterise points.

The projection of a 3D world point into an image is given by $\pi_{\mathbf{K}} : \mathbb{R}^3 \to \Omega$ with inverse projection $\pi_{\mathbf{K}}^{-1} : \Omega \times \mathbb{R}^+ \to \mathbb{R}^3$ which requires the inverse depth of the point. $\mathbf{K}$ denotes the camera's intrinsic parameters for which we use the pinhole model comprising of x- and y-axis focal lengths and image principal point and are assumed to be known.

Poses are represented by rigid-body transformations $\mathbf{T} \in SE(3)$ which transform 3D world points from the one frame to another according to a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. That is,

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \qquad \text{where} \quad \mathbf{R} \in SO(3) \quad \text{and} \quad \mathbf{t} \in \mathbb{R}^3. \tag{5.1}$$

For optimisation, we use a minimal representation of the camera poses given by the element of the associated Lie-algebra $\xi \in \mathfrak{se}(3)$ such that $\mathbf{T} = \exp_{\mathfrak{se}(3)}(\xi)$ and $\xi = \log_{\mathrm{SE}(3)}(\mathbf{T})$. The transformation of a point from the world frame to the camera frame of image i is written as ξ_i whereas the transformation from frame i to j is written as $\xi_{i\to j}$. Poses can be composed with the operator $\circ : \mathfrak{se}(3) \times \mathfrak{se}(3) \to \mathfrak{se}(3)$ such that

$$\xi_{i\to k} := \xi_{i\to j} \circ \xi_{j\to k} := \log_{\mathrm{SE}(3)}(\exp_{\mathfrak{se}(3)}(\xi_{i\to j}) \cdot \exp_{\mathfrak{se}(3)}(\xi_{j\to k})). \tag{5.2}$$

With some abuse of notation, when referring to the general case of the pose between some pair of images I and I', we write the transformation simply as ξ.

5.3.2 Direct visual odometry

Consider two images I and I'. Direct visual odometry seeks to find the motion parameters ξ which minimise the photometric error,

$$\xi^* = \arg\min_{\xi} \sum_{\mathbf{m}\in\Omega} || \underbrace{I(w(\mathbf{m}; D(\mathbf{m}), \Delta\xi)) - I'(w(\mathbf{m}; D(\mathbf{m}), \xi))}_{r(\mathbf{m};d,\xi)} ||_2^2. \tag{5.3}$$

In Equation 5.3, we use the the warp function $w : \Omega \times \mathbb{R} \times \mathfrak{se}(3) \to \Omega$ which warps a point $\mathbf{m} \in \Omega$ in image I to $\mathbf{m}' \in \Omega'$ in image I' according to the estimated rigid-body transformation, inverse depth, and pinhole model projection. That is,

$$\mathbf{m}' = w(\mathbf{m}; d, \xi) = \pi_{\mathbf{K}}(\mathbf{R}\pi_{\mathbf{K}}^{-1}(\mathbf{m}; d) + \mathbf{t}). \tag{5.4}$$

Equation 5.3 is given in the *inverse-compositional* form of the Lucas-Kanade image alignment algorithm which is solved by gradient descent. This form is efficient to solve because the Jacobian matrices $\mathbf{J}$ are calculated at the zero warp and can therefore be pre-computed and remain constant for all iterations.

$$\Delta\xi = \sum_{\mathbf{m}\in\Omega} (\mathbf{J}(\mathbf{m}; D(\mathbf{m}))^T \mathbf{J}(\mathbf{m}; D(\mathbf{m})))^{-1} \mathbf{J}(\mathbf{m}; D(\mathbf{m}))^T \mathbf{r}(\mathbf{m}; D(\mathbf{m}), \xi) \tag{5.5}$$

$$\text{where} \quad \mathbf{J}(\mathbf{m}; d) = \nabla I(\mathbf{m}) \frac{\partial w(\mathbf{m}; d, \xi)}{\partial \xi}\Big|_{\xi=\mathbf{0}} \tag{5.6}$$

The update computed in each iteration is therefore

$$\xi \leftarrow \xi \circ \Delta\xi. \tag{5.7}$$

5.3.3 Multi-channel and descriptor-based Lucas-Kanade

The LK algorithm can operate on an arbitrary number of image channels, N_c, by finding the L_2 or similar norm between each channel separately. This allows us to perform alignment of multi-channel images, such as RGB images, or even using dense descriptor images, such as densely evaluated SIFT descriptors. This gives us the form

$$\xi^* = \arg\min_{\xi} \sum_{\mathbf{m}\in\Omega} \sum_{i=0}^{N_c-1} || \underbrace{\Phi_i(w(\mathbf{m}; D(\mathbf{m}), \Delta\xi)) - \Phi_i'(w(\mathbf{m}; D(\mathbf{m}), \xi))}_{r_i(\mathbf{m};d,\xi)} ||^2 \tag{5.8}$$

where $\Phi_i(\cdot)$ is the value of the ith channel. This gives us a different update, given by

$$\Delta\xi = \sum_{\mathbf{m}\in\Omega} \sum_{i=0}^{N_c-1} (\mathbf{J}_i(\mathbf{m}; D(\mathbf{m}))^T \mathbf{J}_i(\mathbf{m}; D(\mathbf{m})))^{-1} \mathbf{J}_i(\mathbf{m}; D(\mathbf{m}))^T \mathbf{r}_i(\mathbf{m}; D(\mathbf{m}), \xi) \tag{5.9}$$

$$\text{where} \quad \mathbf{J}_i(\mathbf{m}; d) = \frac{\partial \Phi_i}{\partial \mathbf{m}} \frac{\partial w(\mathbf{m}; d, \xi)}{\partial \xi}\Big|_{\xi=\mathbf{0}}. \tag{5.10}$$

5.4 Direct tracking with binary features

In this section, a number of methods for performing IC-LK gradient descent with *binary* descriptors is presented. Firstly, one of the most basic binary descriptors, the Census transform, is introduced. Then, a method for estimating the gradient and descent direction is outlined. This method is extended to arbitrary binary descriptors, and finally, an alternate method based on the Rank transform is proposed.

5.4.1 The Census and Rank transforms

We firstly consider the Census and Rank transforms which are single-channel descriptors invariant to global monotonically increasing rescaling of the image [260]. The Census transform compares a target pixel to each other pixel in a local patch, for example 3×3, and sets a corresponding bit in a bit string to one if it is greater than or equal.

$$\Phi_C(\mathbf{m}) := \{\mathbb{1}_{(\mathbf{m} \geq \mathbf{m}+\Delta\mathbf{m}_0)}, \ldots, \mathbb{1}_{(\mathbf{m} \geq \mathbf{m}+\Delta\mathbf{m}_{N_b})}\}$$
$$\text{with} \quad \mathbb{1}_{(x)} = \begin{cases} 1 & \text{if } x \text{ is true,} \\ 0 & \text{otherwise} \end{cases}$$

where $\{\Delta\mathbf{m}_i\}_{i=0}^{N_b-1}$ is the set of neighbouring pixels within the patch and N_b is the number of bits in the descriptor. The Rank transform is simply the sum of all set bits

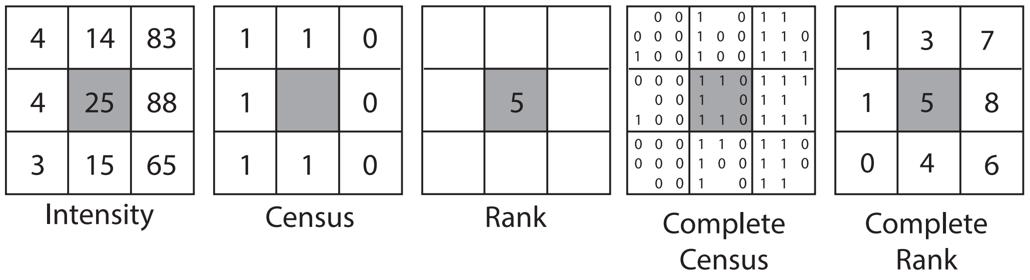


Figure 5.1: Various Census transform-related descriptors shown for an example 3×3 intensity patch. The Census transform uses comparison operations over intensity to encode the local structure into a bit string and the Rank transform is the sum of set bits in the Census transform. The Complete Census is the Census transform repeated with each pixel acting as the base point, and the Complete Rank is the Rank transform applied to each component of the Complete Census.

in the corresponding Census transform. That is, the Rank descriptor is essentially the *Hamming weight* of the corresponding Census transform, given by

$$\Phi_R(\mathbf{m}) := ||\Phi_C(\mathbf{m})||_H$$
$$\text{where} \quad ||\Phi||_H = \sum_{i=0}^{N_b-1} (\Phi)_i$$

with $(\cdot)_i$ referring to the ith bit of the bit-string. We also consider two additional descriptors. The Complete Census computes a Census transform for each pixel in the image patch and the Complete Rank takes the Hamming weight of each of those Census transforms [12]. An example of these four descriptors on a 3×3 intensity patch is shown in Figure 5.1.

5.4.2 Estimating gradients and descent direction

Binary descriptors are matched using the *Hamming distance* norm rather than L_2 norm. This counts the number of bits that differ between two binary descriptors, given by

$$||\Phi_1,\ \Phi_2||_H = \sum_{i=0}^{N_b-1} (\Phi_1 \oplus \Phi_2)_i$$

where $\oplus$ is the *exclusive-or* (XOR) operator. This norm results in a non-convex and non-differentiable cost surface, preventing us from solving Equation 5.8. Bit-planes [265] resolves this by separating each bit of the descriptors into a different channel and performing LK on each binary image, however, this method becomes computationally expensive with larger descriptors which could have as many as 512 channels.

For a single-channel solution, we consider the relationship between Rank and Census transforms. If we were able to minimise Equation 5.8 for Census descriptors,

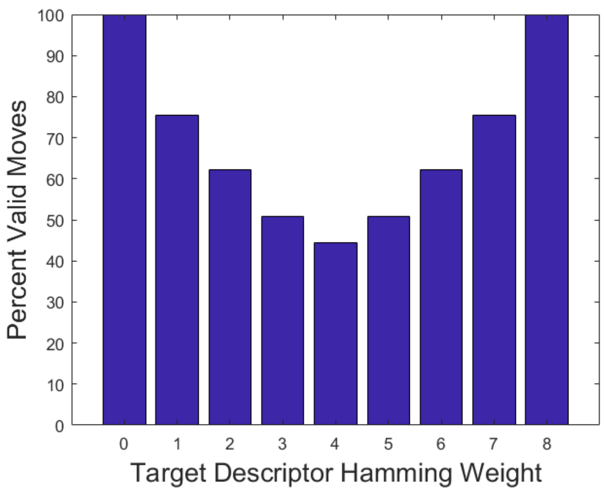


Figure 5.2: For pairs of Census descriptors Φ_1 and Φ_2, the percentage of changes from Φ_2 to some Φ_2' where a reduction in difference of Hamming weights also results in a reduction in Hamming distance, separated by weight of Φ_1.

we would be minimising Hamming *distance*. On the other hand, we can trivially apply Equation 5.8 to an image of Rank descriptors in the same manner as it is applied to raw intensity, using the L_2 norm. In that case, we would essentially be minimising the *difference* between Hamming *weights*. That is, since the Rank transform is the Hamming weight of the corresponding Census transform, the L_2 norm between two Rank descriptors is the difference between Hamming weights of those Census descriptors.

Consider now two Census descriptors:

$$\Phi_1 = \{1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1\}$$
$$\Phi_2 = \{1,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1\}$$

The Hamming distance between these two descriptors is 4, whereas their Hamming weights are 6 and 4, respectively, giving a difference of 2. How consider moving from Φ_2 to a neighbouring descriptor $\Phi_2' = \{1, 1, 0, 0, 1, 0, 1, 1\}$. The Hamming distance has now reduced to 3 and the difference in Hamming weights is 1. In this particular example, a reduction in *difference of Hamming weights* has corresponded to a reduction in Hamming *distance* as well. That is to say, in this case, following the Rank descent also improved Census descriptor cost.

Obviously, this is not universally the case, and one could establish a counter-example where the Rank descent direction corresponds to an increase in Census descriptor cost. Exploring this, Figure 5.2 shows the percentage of movements where reducing difference in Hamming weights also reduces Hamming distances. As can be seen, the two comparisons align for the majority of cases.

The Rank transform alone could potentially be used for tracking with the standard L_2 norm, however, it is far less discriminative than the Census transform. Therefore, we propose to get residual size from the Hamming distance of Census descriptors, but gradient and decent direction from the Rank descriptors. This allows gradient descent to be applied as though using the Rank descriptor, but to retain the relatively more discriminative nature of the Census transform. That is,

$$r(\mathbf{m}; d, \xi) = \mathbb{1}^{\epsilon}_{(\Phi_R(\mathbf{m}),\ \Phi'_R(w(\mathbf{m};d,\xi)))} \cdot ||\Phi_C(\mathbf{m}),\ \Phi'_C(w(\mathbf{m}; d, \xi))||_H \tag{5.11}$$

$$\mathbf{J}(\mathbf{m}; d) = \frac{\partial \Phi_R}{\partial \mathbf{m}} \frac{\partial w(\mathbf{m}; d, \xi)}{\partial \xi}\Big|_{\xi=\mathbf{0}} \tag{5.12}$$

where $\mathbb{1}^{\epsilon}_{(x_1,x_2)}$ is the signed indicator function,

$$\mathbb{1}^{\epsilon}_{(x_1,x_2)} = \begin{cases} 1 & \text{if } x_1 - x_2 > \epsilon, \\ 1 & \text{if } x_2 - x_1 > \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

5.4.3 Extending to arbitrary binary descriptors

The residuals and Jacobians in Equations 5.11 and 5.12 can be extended to a binary descriptor of any size by using Hamming weights and distances in general. That is,

$$r(\mathbf{m}; d, \xi) = \mathbb{1}^{\epsilon}_{(||\Phi(\mathbf{m})||_H,\ ||\Phi'(w(\mathbf{m};d,\xi))||_H)} \cdot ||\Phi(\mathbf{m}),\ \Phi'(w(\mathbf{m}; d, \xi))||_H \tag{5.13}$$

$$\mathbf{J}(\mathbf{m}; d) = \frac{\partial ||\Phi||_H}{\partial \mathbf{m}} \frac{\partial w(\mathbf{m}; d, \xi)}{\partial \xi}\Big|_{\xi=\mathbf{0}}. \tag{5.14}$$

Therefore, the approach when applied to general binary descriptors can be summarised as follows:

- Residual magnitude is determined by the Hamming *norm* between descriptors,
- Descent direction is determined by a signed indicator function applied to the Hamming *weights* of each descriptor,
- Corresponding to the image gradient in Equation 5.6, we use the gradient of Hamming *weights* of the unwarped descriptor image.

For dense descriptor images to work with gradient descent, they need good convergence basins which are the region around the correct pixel match that results in convergence. To demonstrate a reasonable convergence basin for the descriptors we are using, we show in Figure 5.3 the sum of squared difference and sum of Hamming distance costs over a translated window, which show a clear basin leading to the correct translation at (0, 0).

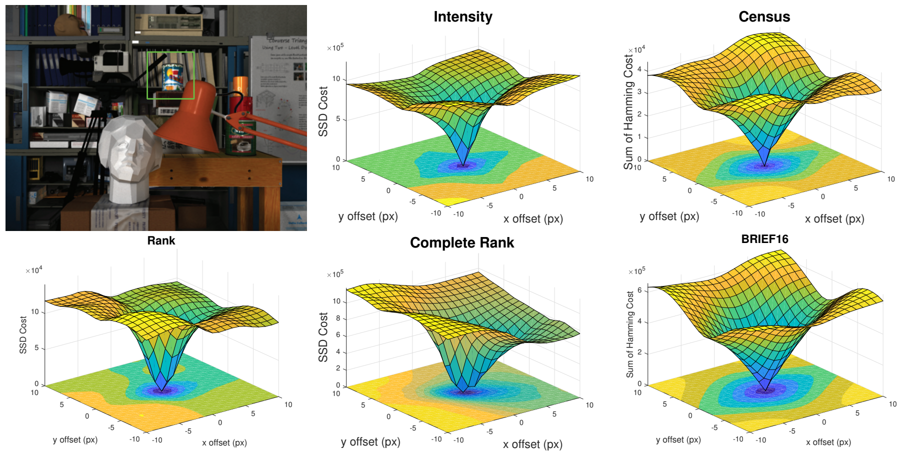


Figure 5.3: The cost surface of a translated section of the image. The highlighted region is compared to a translated region using various descriptors and their norms. Sum of squared difference is used for intensity, Rank and Complete Rank. Sum of Hamming distances is used for Census and BRIEF descriptors.

5.4.4 Rank approximations

In the same vein as tracking directly on Rank transform descriptors, we consider the possibility of tracking entirely on the Hamming weight of a binary descriptor. If one computes a dense image of binary descriptors then takes the Hamming weight of all descriptors, the resulting image is directly suitable for tracking with the L_2 norm with Equation 5.3. Expressing the residual in terms of the binary descriptor itself, this would still use the Jacobian of Equation 5.14 but replace the corresponding residual with Equation 5.15. We could also explore this in a multi-channel manner, expressed in Equations 5.16 and 5.17, where each n-bit block of the descriptor is a separate channel, for example 8-bit channels. Although this reduces structural discrimination it maintains invariance to global monotonically increasing rescaling of intensity.

$$r(\mathbf{m}; d, \xi) = ||\Phi(\mathbf{m})||_H - ||\Phi'(w(\mathbf{m}; d, \xi))||_H \tag{5.15}$$

$$r_i(\mathbf{m}; d, \xi) = ||\Phi_i(\mathbf{m})||_H - ||\Phi'_i(w(\mathbf{m}; d, \xi))||_H \tag{5.16}$$

$$\mathbf{J}_i(\mathbf{m}; d) = \frac{\partial ||\Phi_i||_H}{\partial \mathbf{m}} \frac{\partial w(\mathbf{m}; d, \xi)}{\partial \xi}\bigg|_{\xi=\mathbf{0}} \tag{5.17}$$

5.4.5 Implementation details

The binary descriptor-based IC-LK tracking was implemented in C++ using the OpenCV and Eigen libraries. Points were selected in each image using local maximums within a 3×3 patch with non-zero gradients. We tracked on a four-level Gaussian pyramid and built a depth map for the point warping from small baseline epipolar searches between images of known relative pose. For intensity, we used the

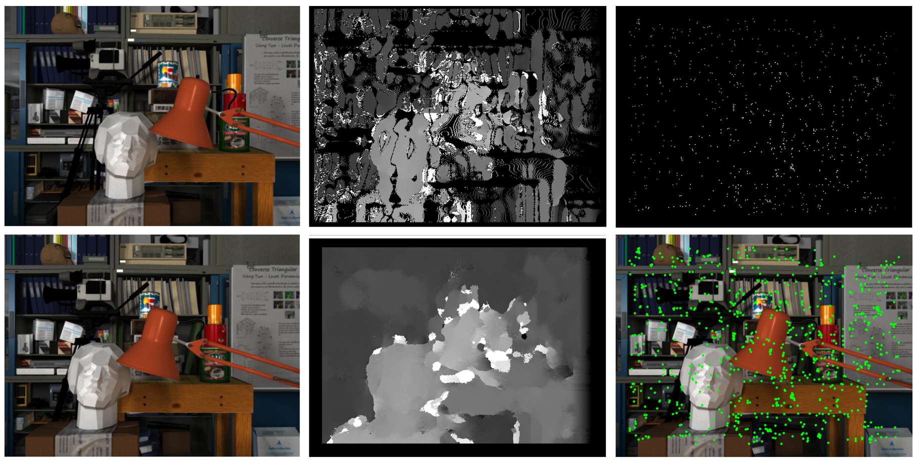


Figure 5.4: Example stereo pair from New Tsukuba dataset [267]. Middle column has dense depth maps built from scanning epipolar lines for minimum SSD over a pixel patch (top), and scanning epipolar lines for best BRIEF16 descriptor match (bottom). Right column shows the sparse depth map with BRIEF16 descriptors (top) and successful points overlaid on image (bottom).

method of LSD-SLAM [92], searching the epipolar line for the minimum SSD over a patch. For the binary features we replaced the patch-based SSD with Hamming distance between descriptors. This process is shown in Figure 5.4. This method requires bootstrapping for the first depth map or pose, as each require the other. We used a standard feature-based method to estimate the first pose then proceeded as described [257].

5.5 Evaluations

We evaluated our proposed methods using the KITTI Visual Odometry dataset seen in Figure 5.5 [268]. This is comprised of footage taken on a road in a residential area.

Firstly, we tested the method of Section 5.4.2 where residual magnitude was determined by Census descriptors but gradient and descent direction was determined by Rank transform. This was compared to tracking using Rank and Compete Rank alone, as well as to ground truth, raw intensity, and Bitplanes [265]. Figure 5.6 shows on these results on the left, with intensity having the poorest performance. The Rank transform performs better than intensity and then our approach with the Census transform improves upon Rank slightly. This shows the effect of using the more discriminative Census transform, however, its effect with these simple descriptors is only minor. The Bitplanes method is much better than our method applied to the Census transform, however, it performs on par with simply tracking on the Complete Rank descriptor.

We then tested our approach applied to the BRIEF16 descriptor, as well as the

Figure 5.5: Example images from the KITTI dataset used for evaluation. Frame 0 (top) and frame 200 (bottom) from sequence 00 [268].

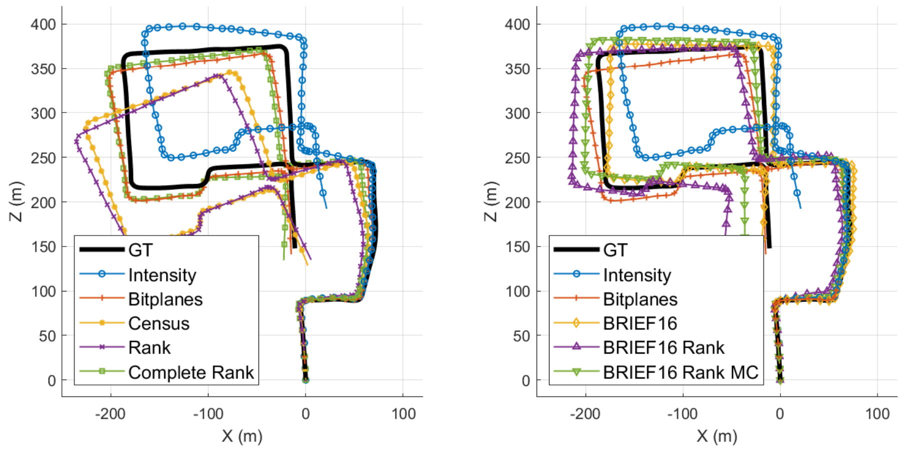


Figure 5.6: Estimated paths for the first 1500 frames of the test dataset, using intensity, Bitplanes, Census, Rank, Complete Rank, BRIEF16, and Hamming weight approximations of BRIEF16.

single- and multi-channel Rank approximation alternatives. Again, this was compared to ground truth, raw intensity, and Bitplanes. Figure 5.6 shows on these results on the right. Our primary method improved upon the Bitplanes method when applied to more robust descriptors. The multi-channel Rank approximation method also had excellent performance, noticeably below our primary method but better than the single-channel Rank approximation method.

Following this, we tested BRIEF16, BREIF32, ORB and BRISK descriptors with our primary method of Section 5.4.3, shown compared to the ground truth in Figure 5.7. It is clear from this comparison that all four robust binary descriptors performed to a similar accuracy, however, overall the ORB descriptor was the most accurate. The other three descriptor types performed extremely similarly and would require further testing on a wider variety of datasets to clearly determine a difference.

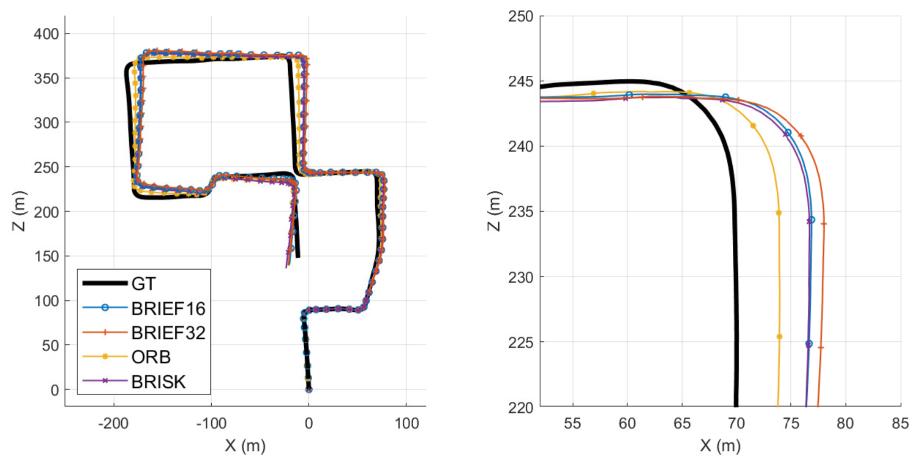


Figure 5.7: Estimated paths for the first 1500 frames of the test dataset, using intensity, BRIEF16, BRIEF32, ORB, and BRISK descriptors. A zoomed in section is shown on the right.

Overall, these comparisons reveal a range of interesting information. Tracking on intensity alone does not perform very well, however Bitplanes performs much better. Our single-channel Census method is slightly better than a Rank transform approach but performs below Bitplanes. Interestingly, the Complete Rank transform performs almost as well as Bitplanes. The more complex descriptors perform much better than the prior types, with all such descriptors performing similarly. However, approximating the entire descriptor with its Hamming weight, even in a multi-channel case, sees poorer performance.

Comparing the frame-by-frame error of BRIEF16 to Bitplanes in Figure 5.8, we can see the very similar but clearly improved accuracy. This is because, although our descent direction is only an approximation, BRIEF descriptors provide much more information to exploit compared to Census descriptors. Therefore, our method can be seen as a trade-off: approximating the gradient and descent direction reduces the accuracy compared to the Bitplanes method, however, by enabling the use of more robust descriptors in an efficient manner, even greater gains in accuracy are achieved.

Looking at the ranges of errors for each descriptor type across all frames in Figure 5.9, we can see that our methods for BRIEF, ORB and BRISK out-perform the other methods quite substantially, with ORB being marginally better than BRIEF16. Interestingly, the BRIEF32 descriptor performed worse than the BRIEF16 descriptor, which warrants further investigation.

Table 5.1 shows the channels and bytes per channel for each descriptor and the overall root-mean-squared error for translation and rotation, with ORB being most accurate for all measurements. ORB was much better than the alternative descriptor types for translation, however, this difference was more minor for rotation. The

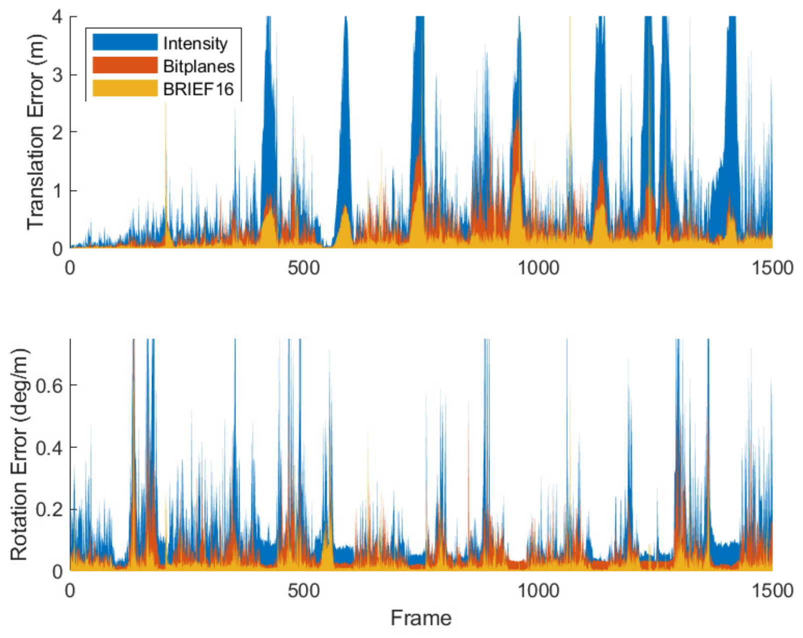


Figure 5.8: Error per frame in estimations of relative poses, shown for intensity, Bitplanes, and BRIEF16 descriptors.

multi-channel approximate method for BRIEF16 can be seen to have about twice the error than our primary method when applied to the same descriptor type. From only an error point-of-view, we can clearly recommend the ORB descriptor as the best out of the alternatives we have considered.

Considering processing time instead, Table 5.2 shows the execution times of different parts of the algorithm for each of the four robust descriptors as well as raw intensity. This includes constructing the Gaussian pyramids, computing the descriptors and gradients, calculating the Jacobian, and performing one iteration of the IC-LK alignment. The raw intensity method does not require any descriptor calculation. In this comparison, we considered the case of computing dense descriptor images as well as only computing the sparse keypoints that are actually utilised. The naïve implementation computing dense descriptor images is clearly not suitable for real-time use, however, intelligently computing the minimal number of descriptors that get used speeds this up substantially. Much more processing time is spent in the robust descriptor methods, however, in the sparse case this is still within reason for real-time application. Of the alternatives considered, the BRIEF16 descriptor was the fastest which is understandable as it is the shortest descriptor. When comparing both accuracy and execution time, this demonstrates that BRIEF16 is an excellent choice for our method.

Table 5.3 compares the results of our method using the complex descriptors to other popular methods on the KITTI dataset. DSO [93] is a popular monocular

Table 5.1: Number of channels, bytes per channel, and RMSE for translation and rotation.

Descriptor	Channels	Bytes	X(m)	Y(m)	Z(m)	Rotation(deg/m)
Intensity	1	1	0.814	1.342	0.655	0.272
Bitplanes [265]	8	1	0.508	0.413	0.231	0.147
Census [260]	1	1	0.966	0.291	0.551	0.133
Rank [260]	1	1	1.136	0.415	0.725	0.115
Complete Rank [258]	9	1	0.519	0.845	0.318	0.193
BRIEF16 [152]	1	16	0.311	0.330	0.236	0.059
BRIEF16 Rank	1	1	1.044	0.910	0.617	0.118
BRIEF16 Rank MC	16	1	0.643	0.409	0.318	0.105
BRIEF32 [152]	1	32	0.360	0.373	0.273	0.068
ORB [101]	1	32	**0.270**	**0.304**	**0.173**	**0.057**
BRISK [151]	1	64	0.379	0.399	0.258	0.078

Table 5.2: Execution time for different parts of the algorithm for raw intensity and four descriptor types. Times given in milliseconds (ms).

Stage	RI	BRIEF16	BRIEF32	ORB	BRISK
Pyramid Construction			0.195		
Descriptor Calculation (Dense)	N/A	120.93	265.67	58.74	636.80
Gradient Calculation (Dense)	0.25	13.87	13.27	13.88	13.41
Descriptor Calculation (Sparse)	N/A	3.03	3.27	2.97	85.05
Gradient Calculation (Sparse)	0.03	0.46	0.43	0.48	0.43
Jacobian Calculation	0.46	0.70	0.99	1.09	1.23
LK Iteration	0.52	1.11	1.59	1.87	2.58

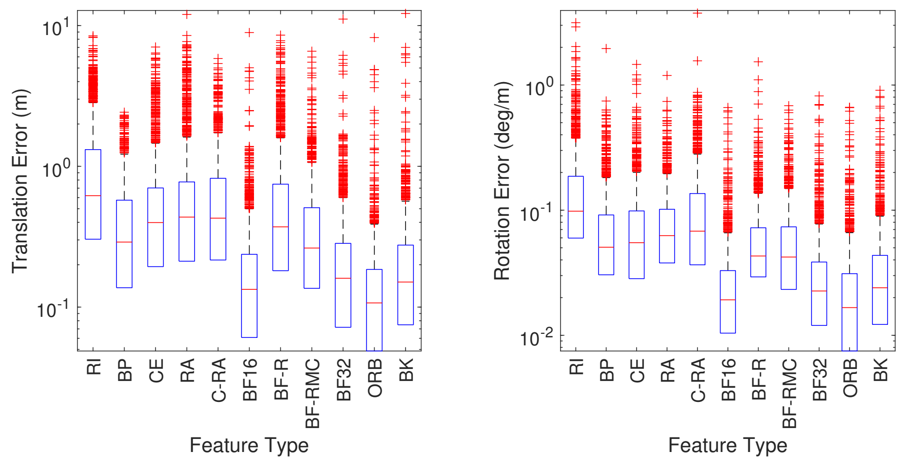


Figure 5.9: Translation and rotation errors for relative pose estimations between frames, for raw intensity, Bitplanes, Census, Rank, Complete Rank, BRIEF16 with single and multi-channel approximations, BRIEF32, ORB, and BRISK descriptors.

Table 5.3: Comparison of our method on the KITTI dataset with four types of descriptors to two widely used monocular methods and their stereo counterparts.

Method	Translation (%)	Rotation(deg/m)
Ours BRIEF16	7.80	0.059
Ours BRIEF32	7.84	0.068
Ours ORB	7.59	0.057
Ours BRISK	10.89	0.078
Mono DSO [93]	9.82	0.0021
ORB-SLAM [100]	14.37	0.0029
Stereo DSO [108]	0.93	0.0020
ORB-SLAM2 [109]	1.15	0.0027

visual odometry method and ORB-SLAM [100] is a monocular SLAM method. Both have better results in terms of rotation whereas our results outperform in terms of translation. Comparatively, Stereo DSO [108] and ORB-SLAM2 [109] demonstrate the performance gains from introducing a second camera for stereo matching, with significantly more accurate results for translation. We believe these results demonstrate that our method provides a novel new approach to monocular visual odometry with strong potential for further development.

5.6 Summary

This chapter has proposed a single-robot visual odometry method that is suitable for use in the second stage of the distributed calibration, localisation, and mapping pipeline presented in Figures 1.2 and 2.28. The primary goal of this method was to incorporate the robust binary feature descriptors often used for wide-baseline matching between robots into the highly efficient direct visual odometry method.

Binary feature-based direct tracking shows promising performance compared to intensity and alternative feature-based direct methods. Our method approximates gradient and descent direction with Hamming weights, allowing the descriptors to still be compared using Hamming distance. We derived this by considering the relationship between Census and Rank transforms and extending to more complex descriptors. Compared to alternative approaches for using binary descriptors in the LK algorithm, ours uses a single channel regardless of descriptor size. We also described further single- and multi-channel approximations using Hamming weights entirely. On Census transforms, our method has similar performance to intensity, but on BRIEF, ORB and BRISK descriptors we have much improved performance. Our alternative approximations also perform better than intensity but worse than our primary method.

We plan to integrate this type of binary descriptor tracking into existing VO and SLAM methods to thoroughly demonstrate its performance. Also, we will seek to improve the speed of the algorithm by intelligently computing the minimum number of sparse descriptors required in an improve 'lazy' manner, which our results showed is necessary for real-time execution. We can further improve speed with single-instruction-multiple-data (SIMD) optimisations. Finally, as the features we use are suitable for wide-baseline matching, our method is aimed to be extended into a multi-camera VO system.

From this, the issues raised regarding the efficiency and use of robust descriptors for the local tracking stage of the proposed pipeline have been addressed.

Chapter 6

Optimising edge weights for distributed inference with Gaussian belief propagation[a]

Contents

w

6.1 Summary of contributions

- Two empirically derived methods are presented for the determination of edge-weights in Gaussian belief propagation.
- The weights are determined through the analysis of small-world networks of varying sizes.

[a]Chapter adapted from "B. Halloran *et al.*, 'Optimizing edge weights for distributed inference with gaussian belief propagation,' in *Proceedings of ICIC 2018: Intelligent Computing Theories and Application*, D.-S. Huang *et al.*, Eds., Springer International Publishing, 2018, pp. 46–59, ISBN: 978-3-319-95930-6. DOI: 10.1007/978-3-319-95930-6_6"

- The first method uses uniform edge-weights across the network based on the average node degree.
- The second method uses different weights for each edge based on the average of node degrees either side of an edge, therefore only using information known locally and from direct neighbours.
- The improvements in convergence speed and accuracy of the estimation are demonstrated on a simplified factor graph localisation problem.

6.2 Introduction

In the continuation of contributions to the proposed pipeline in Figures 1.2 and 2.28, this chapter considers the performance of the distributed algorithms that would be utilised for global pose optimisation in stages one and three.

Distributed processing is becoming more of an important paradigm as larger networks are being applied to various problems, particularly robotics and robotic vision. Traditional centralised solutions, where a single central processor takes data from each node, suffer from scalability issues due to communication bottle-necks and have a substantial risk of a single point of failure at the central node. Distributed algorithms alleviate these issues by moving the processing to the nodes themselves and using only local information with communication between direct neighbours, thereby taking better advantage of the communication structure and increasing scalability. These algorithms still need improvements, however, to achieve the same accuracy as centralised systems.

Belief propagation (BP) is a well-known message passing algorithm for performing efficient inference on graphical models [197]. Whilst the original BP algorithm performed inference on discrete random variables, Gaussian belief propagation (GaBP) is an extension of BP to continuous random variables which are modelled as Gaussian densities [200, 242]. This allows many problems dealing with continuous variables, such as localisation, to be interpreted as a distributed problem on a graph.

This chapter proposes improvements to the accuracy and convergence speed of Gaussian belief propagation through the choices of edge-weights based on the analysis of small-world networks.

6.2.1 Related Work

Belief propagation was first proposed by Pearl for inference over discrete random variables, guaranteeing exact inference on acyclic graphs [197]. It was later reformulated as Generalized Belief Propagation (GBP) for approximate inference on cyclic graphs [199]. It has also been shown that BP is equivalent to many similar sum-product algorithms, such as fast Fourier transforms, Kalman filtering, and average-consensus [94].

BP has been extended to work on continuous variables through *non-parametric belief propagation* (NBP) [210], which uses particle filtering, and *Gaussian belief propagation* (GaBP) [200, 242], which uses Gaussian densities. Although NBP can represent more complex non-Gaussian distributions, GaBP requires much less memory and processing power making it more suitable for low-cost hardware. GaBP has been applied to many problems such as parallel SVMs [213], MMSE estimators [269], Kalman filtering [214], linear least squares estimation [270], and multi-camera calibration [36]. NBP has also seen successful application to localisation and tracking [271, 272], however GaBP has been shown to achieve similar results through iterative linearisation of the problem [273]. Recently, GaBP has been applied to finite-element analysis, allowing greater parallelisation on GPUs [274] and high-performance computing applications [275]. Recent work has been done to understand the convergence conditions of GaBP, particularly analysing the information matrix [208, 209]. Other work has sought to improve the accuracy of the different BP algorithms, generally using different strategies for applying importance weights to the messages [210] Tree-reweighted BP (TRW-BP) calculates the entire set of possible spanning-trees for a given cyclic graph then weights edges based on the probability that an edge appears in the set, calling these edge appearance probabilities (EAPs) [276]. It was later shown that uniform weights across all edges could achieve similar results by approximating the EAPs [277]. This has been successfully applied to NBP (TRW-NBP), however the empirical relationship derived was based only on a specific network size [211]. Consequently, we are motivated to derive a more general weighting system for networks of varied sizes and connectivities.

6.2.2 Contributions

To increase the accuracy of Gaussian belief propagation, we present the following improvements. Firstly, we have derived uniform message weights for minimising root-mean-square error (RMSE), approximating the EAPs, by analysing optimal weights for a range of small-world networks of differing sizes and connectivity. Next, we have proposed an alternative non-uniform message weight based on our prior derivation, which provides a further improvement to the algorithm accuracy. Finally, we have provided a practical example of this being used in a distributed inference problem through a demonstration of distributed localisation. The motivation for this is to improve the convergence speed and accuracy of the distributed algorithms that are utilised in the calibration, localisation, and mapping pipeline that has been

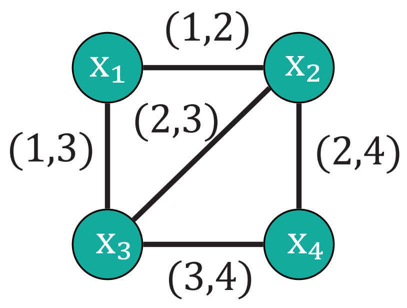


Figure 6.1: An example Markov random field, with nodes and edges identified.

discussed throughout this dissertation.

The remainder of the chapter is organised as follows: Section 6.3 reviews the preliminaries of undirected graphical models and belief propagation; Section 6.4 describes GaBP and reweighted messages; Section 6.5 explains our empirical approach to optimal weights; Section 6.6 provides an application of this algorithm to the localisation problem; Section 6.7 applies the algorithm to the distributed calibration and localisation discussed in Chapters 3 and 4; and finally, Section 6.8 concludes this chapter.

6.3 Preliminaries

In Section 2.5.1 the concept of the undirected graph was introduced, and in Section 2.5.2 the belief propagation algorithm was discussed. However, in the following section, the important aspects of these preliminary concepts are reintroduced for clarity.

As in previous chapters, consider a network of N nodes which are connected in an ad-hoc manner. This network can be described by an undirected graph, or *Markov random field* (MRF), $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Here, $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges containing $(i,j) \in \mathcal{E}$ pairs of nodes where node i and j are connected. We denote the direct neighbours of node i as $\mathcal{N}_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$, which has degree $d_i = |\mathcal{N}_i|$. An example of a simple MRF is shown in Figure 6.1.

Consider each node i having a true state given by some variable z_i. The measurement x_{ij} is the noisy observation made by node i of either z_j itself, or some constraint on z_j, where $j \in \mathcal{N}_i$. We wish to to estimate z_i from all x_{ij} by considering the joint density in Equation 6.1 with node potentials ϕ_i and edge potentials ψ_{ij}. This is solved using belief propagation by passing messages $m^{(t)}_{i \to j}(z_j)$, given in Equation 6.2, from node i to node j across edges $(i,j) \in \mathcal{E}$ at every iteration t. The belief at each iteration, given in Equation 6.3 converges towards a global estimate for z_i, resulting in exact inference on trees and approximate inference otherwise. Section 3.6 outlines the relationship between the problem formulation and belief propagation in greater detail.

$$p(z_0, ..., z_{N-1}) \propto \prod_{i \in \mathcal{V}} \phi_i(z_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(z_i, z_j) \tag{6.1}$$

$$m_{i \to j}^{(t)}(z_j) \propto \int_{z_i} \phi_i(z_i) \psi_{ij}(z_i, z_j) \prod_{k \in \mathcal{N}_i \backslash j} m_{k \to i}^{(t-1)}(z_i) dz_i \tag{6.2}$$

$$b_i^{(t)}(z_i) \propto \phi_i(z_i) \prod_{j \in \mathcal{N}_i} m_{j \to i}^{(t)}(z_i) \tag{6.3}$$

6.4 Gaussian belief propagation

In Gaussian belief propagation, random variables are modelled as Gaussian densities. Therefore, two scalars each represent potentials, messages, and beliefs — mean and precision. In this chapter, compared to Chapters 3 and 4, a slightly different form of the messages and beliefs is used based on the construction of an information matrix and measurement vector. The mean is the noisy measurement x_{ij} and the precision P_{ij} is the inverse of its variance $\Sigma_{x_{ij}}$. The graph is then represented by an information matrix Ω, encoding the edges and precisions, and a measurement vector ξ, containing precision-weighted sums of measurements. For example, if each node has relative measurements of its neighbours, $x_{ij} = z_j - z_i + w_{ij}$, as well as an absolute measurement, $x_{ii} = z_i + w_{ii}$, where w is some zero-mean additive Gaussian noise, then we would have

$$\Omega_{ij} = \begin{cases} (d_i + 1)P_{ij} & \text{for } i = j, \\ -P_{ij} & \text{for } j \in \mathcal{N}_i, \\ 0 & \text{otherwise} \end{cases}$$

$$\xi_i = (d_i + 1)x_{ii}P_{ii} - \sum_{j \in \mathcal{N}_i} x_{ij}P_{ij}.$$

We can then reduce the messages of Equation 6.2 to the form used by Bickson

[242] with mean and precision values,

$$\begin{aligned}
P_{i\backslash j}^{(t)} &= \Omega_{ii} + \sum_{k\in\mathcal{N}_i\backslash j} P_{k\to i}^{(t-1)} \\
\mu_{i\backslash j}^{(t)} &= (P_{i\backslash j}^{(t)})^{-1}(\xi_i + \sum_{k\in\mathcal{N}_i\backslash j} P_{k\to i}^{(t-1)}\mu_{k\to i}^{(t-1)}) \\
P_{i\to j}^{(t)} &= -\Omega_{ij}^2(P_{i\backslash j}^{(t)})^{-1} \\
\mu_{i\to j}^{(t)} &= -\Omega_{ii}\mu_{i\backslash j}^{(t)}(P_{i\backslash j}^{(t)})^{-1}
\end{aligned}$$

and the beliefs have the form

$$\begin{aligned}
P_i^{(t)} &= \Omega_{ii} + \sum_{j\in\mathcal{N}_i} P_{j\to i}^{(t)} \\
\mu_i^{(t)} &= (P_i^{(t)})^{-1}(\xi_i + \sum_{j\in\mathcal{N}_i} P_{j\to i}^{(t)}\mu_{j\to i}^{(t)})
\end{aligned}$$

with $\mu_i^{(t)}$ converging towards a global consensus value for z_i.

6.4.1 Message reweighting

Wainwright et al. first showed how messages can be reweighted in tree-reweighted belief propagation, where weights were based on the probability that an edge appears on a spanning-tree subgraph, ρ_{ij} [276]. This method reduces RMSE in the solution but isn't suitable for use in distributed networks as it requires all possible spanning-trees to be determined to calculate the edge appearance probabilities. Savic et al. simplified this by approximating EAPs for lattice graphs and applied this to nonparametric belief propagation [211]. Here, we will apply these weights to the Gaussian-based equations. Firstly, edge weights ρ_{ij} are applied to the messages for belief in Equation 6.3,

$$b_i^{(t)}(z_i) \propto \phi_i(z_i) \prod_{j\in\mathcal{N}_i} (m_{j\to i}^{(t)}(z_i))^{(\rho_{ij})}.$$

Then outgoing messages of Equation 6.2 include this new form of the reweighted belief with the previous message *from* the target node removed, and have the inverse weight applied to the edge potential, giving

$$m_{i\to j}^{(t)}(z_j) \propto \int_{z_i} (\psi_{ij}(z_i, z_j))^{(1/\rho_{ij})} \frac{b_i^{(t-1)}(z_i)}{m_{j\to i}^{(t-1)}(z_i)} dz_i.$$

This can then be converted to mean and precision for the Gaussian beliefs and messages.

$$\begin{aligned}
P_i^{(t)} &= \Omega_{ii} + \sum_{j \in \mathcal{N}_i} \rho_{ij} P_{j \to i}^{(t)} \\
\mu_i^{(t)} &= (P_i^{(t)})^{-1}(\xi_i + \sum_{j \in \mathcal{N}_i} \rho_{ij} P_{j \to i}^{(t)} \mu_{j \to i}^{(t)}) \\
P_{i \backslash j}^{(t)} &= P_i^{(t-1)} - P_{j \to i}^{(t-1)} \\
\mu_{i \backslash j}^{(t)} &= (P_{i \backslash j}^{(t)})^{-1}(P_i^{(t-1)} \mu_i^{(t-1)} - P_{j \to i}^{(t-1)} \mu_{j \to i}^{(t-1)}) \\
P_{i \to j}^{(t)} &= -\Omega_{ij}^2 (\rho_{ij}^2 P_{i \backslash j}^{(t)})^{-1} \\
\mu_{i \to j}^{(t)} &= -\Omega_{ii} \mu_{i \backslash j}^{(t)} (\rho_{ij} P_{i \backslash j}^{(t)})^{-1}
\end{aligned}$$

In TRW-NBP, Savic et al. used uniform weights for the entire graph. That is, $\rho_{ij} = \rho_{opt}$ for all edges $(i, j) \in \mathcal{E}$, with ρ_{opt} being determined by their empirical formula [211].

6.5 Optimising weights for small-world networks

For TRW-NBP, Savic et al. determined an empirical relationship between the optimal edge weights, with regard to RMSE, and the average node degree of the graph n_d, given by

$$\rho_{opt}(n_d) = \rho_0 e^{-k_\rho n_d} \tag{6.4}$$

where ρ_0 and k_ρ were constants. These constants were found empirically for a network of $N = 25$ nodes and were different for lattice and random graphs. In this section, optimal weights are analysed for minimising RMSE in GaBP and extended to more general small-world networks for a range of network sizes.

6.5.1 Small-world generation

A small-world network is an interpolation between lattice graphs, where nodes are connected in a regular pattern, and random graphs. Small-world networks are characterised by high clustering and having few hops between any two nodes. We chose to optimise our algorithm weights for small-world networks due to their prevalence in many naturally occurring systems. These networks can be generated with the Watts-Strogatz model, where a lattice graph has its edges re-wired randomly with some probability [278]. This process is as follows:

1. For a graph of N nodes, it has connectivity or average node degree n_d with $N \gg n_d$ and a parameter representing the 'randomness' of the graph $0 \leq \beta \leq 1$.

2. The graph is initially connected as in a regular lattice where each node is connected to the $n_d/2$ neighbours either side. That is,

$$\mathcal{E} = \{(i,\ j) : 0 < |i - j| \operatorname{mod}(N - \frac{n_d}{2} - 1) < \frac{n_d}{2}\}$$

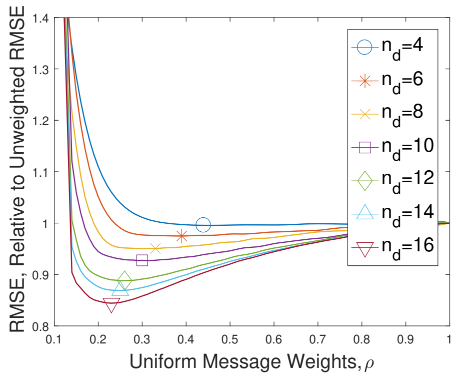
3. For each node i, take each edge (i, j) where $j \in \mathcal{N}_i$ and $i < j$, and with probability β replace it with a random new edge (i, k) where $k \neq i$ and $k \notin \mathcal{N}_i$.

6.5.2 Uniform optimal weights

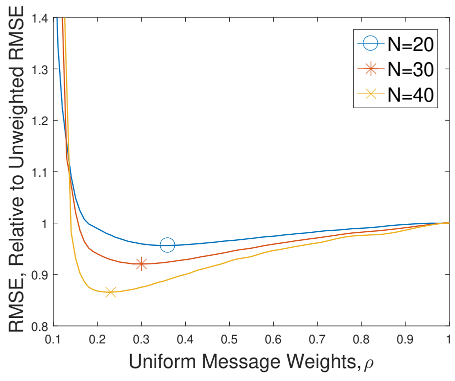
The main effect of the graph cycles on the GaBP solution is an increase of RMSE, with acyclic graphs performing exact inference. Using small-world networks, the effect of uniform message weights on RMSE was simulated on a variety of structures, with four network sizes N and β varying from 0 to 1 in steps of 0.1. For $N = 10$, small-worlds were generated of connectivities $1 \leq n_d/2 \leq 4$. For $N = 20$ and $N = 30$, connectivities were $1 \leq n_d/2 \leq 7$. For $N = 40$, connectivities were $2 \leq n_d/2 \leq 8$. For each size, β, and connectivity, 1000 small-world networks were generated. In each network a simple single variable problem was simulated with nodes assigned true values $z_i \in [1,\ 100]$ and measurements taken of themselves and neighbours affected by zero mean Gaussian noise of variance $\Sigma_{x_{ij}} = 0.5$. The maximum likelihood of z_i was estimated using GaBP with uniform weights ρ ranging from 0.01 to 1.0 in steps of 0.01, and the RMSE of these solutions was compared to the RMSE of the unweighted GaBP solution.

Figure 6.2a shows results for $N = 40$ and $\beta = 0.5$. As the connectivity increases the optimal weight decreases, whilst its effect on RMSE increases. For higher connectivities, most weights have some positive effect on the RMSE compared to the unweighted solution. It is only once the weights become much lower than the optimal weight that the error begins to increase rapidly, and the solution no longer converges. Figure 6.2b shows that the location of the optimal weight does not only rely on the average node degree n_d, but also the graph size N. This is because the amount by which the connectivity of a cyclic graph exceeds that of a valid spanning-tree is dependent on N as well. Figure 6.2c shows the effect that different β values in the small-world generation have on the location of the optimal weight. In a network of $N = 40$ and $n_d = 12$, the optimal weights for each value of β cluster around $\rho = 0.25$ with no significant pattern. This result was similar for all graph sizes and connectivities that were tested.

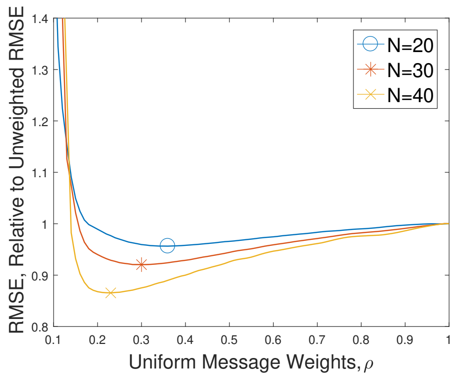
To explore these relationships further, the optimal weights for each graph size and connectivity were analysed against n_d/N. This revealed a power relationship, given



(a) Different connectivities for $N = 40$ and $\beta = 0.5$.



(b) Different network sizes for $n_d = 12$ and $\beta = 0.5$.



(c) Different β values for $N = 40$ and $n_d = 12$.

Figure 6.2: Average RMSE relative to unweighted RMSE for different uniform message weights in small-world networks with different connectivities, network sizes, and β.

in Equation 6.4, as opposed to the exponential relationship of Equation 6.4.

$$\rho_{opt}(n_d, N) = \rho_0(\frac{n_d}{N})^{-k_\rho} \tag{6.5}$$

Figure 6.3 shows this for $N = 40$, taking the average optimal weight across all β. This was repeated for all network stated sizes and revealed changing values of ρ_0 and k_ρ with N, shown in Figure 6.4. Using these trends, the following empirical relationships were derived,

$$\rho_0(N) = a_\rho N^{-b_\rho}$$
$$k_\rho(N) = a_k N + b_k$$

with the resulting data producing $a_\rho = 2.9969$, $b_\rho = 0.845$, $a_k = 0.0026$, and $b_k = 0.444$.

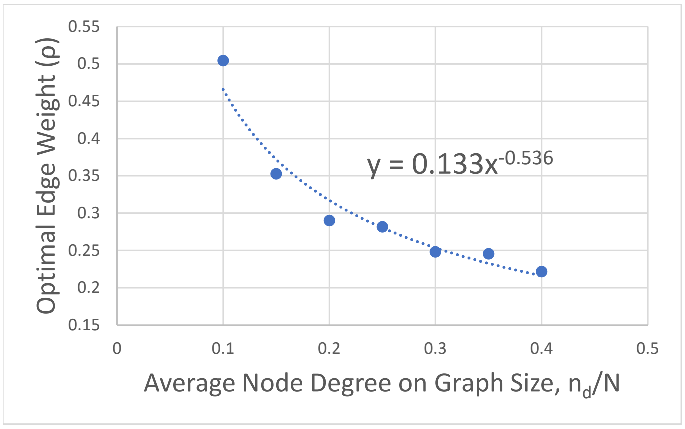
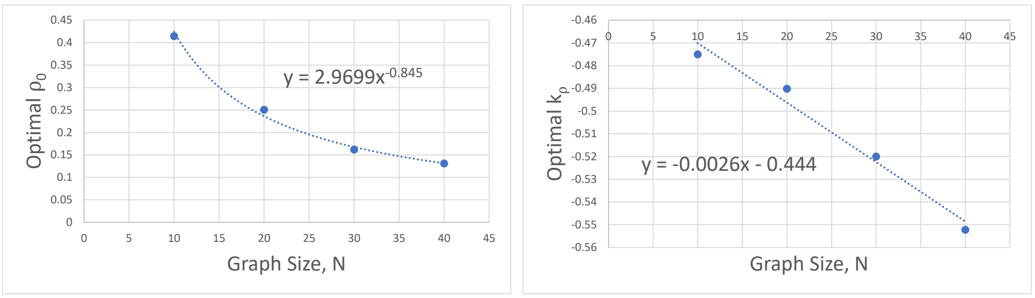


Figure 6.3: Optimal weights for different connectivities against n_d/N, for a network of $N = 40$.



(a) Optimal ρ_0 for different graph sizes. **(b)** Optimal k_ρ for different graph sizes.

Figure 6.4: Optimal values for ρ_0 and k_ρ for different graph sizes.

6.5.3 Non-uniform weights

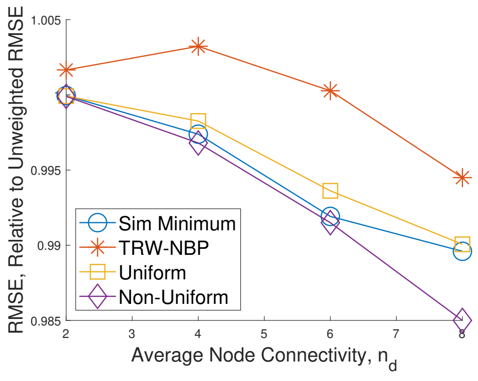
Next, the effect of non-uniform weights was explored. Considering that the heuristic of Equation 6.5 aims to emulate the effect of EAP weights, individual node degrees can be used rather than the average node degree. Since the probability of a given edge being pruned to make a spanning-tree is related to the degrees of the nodes at either end, we use the average of these degrees, d_{ij}, for edge specific weights given in Equation 6.

$$d_{ij} = \frac{d_i + d_j}{2} \tag{6.6}$$

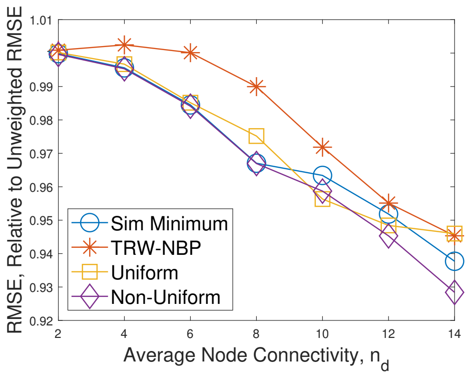
$$\rho_{opt}(d_{ij}, N) = \rho_0(\frac{d_{ij}}{N})^{-k_\rho} \tag{6.7}$$

6.5.4 Comparison of the weighting methods

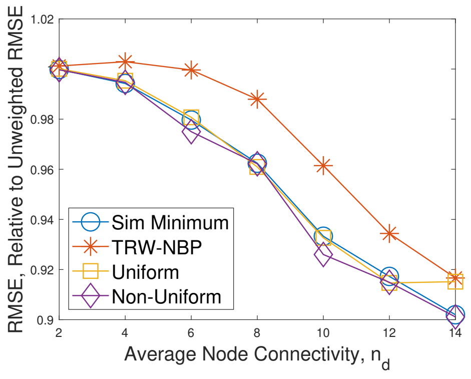
The effects of these different weighting methods were compared using the same rages of graph sizes, connectivities and β as in the previous simulation, measuring the average RMSE relative to the unweighted solution, shown in Figure 6.5. As a baseline, the ideal weights at the minima of the previous simulation, shown in



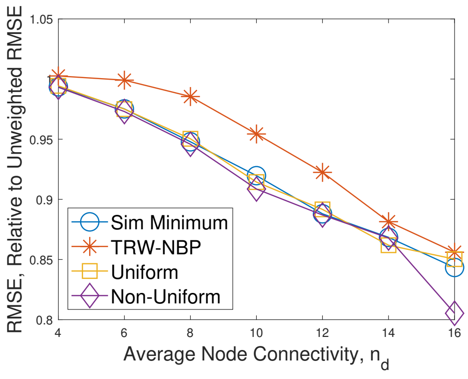
(a) Graph size $N = 10$.



(b) Graph size $N = 20$.



(c) Graph size $N = 30$.



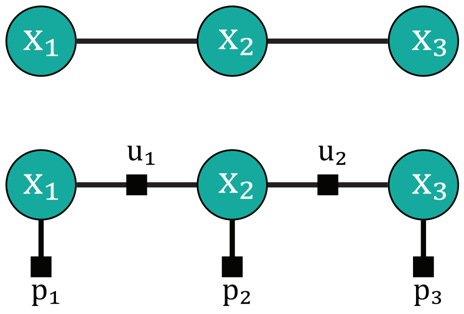
(d) Graph size $N = 40$.

Figure 6.5: Average RMSE, relative to RMSE of unweighted GaBP of 1000 trials, for different weighting systems and graph sizes.

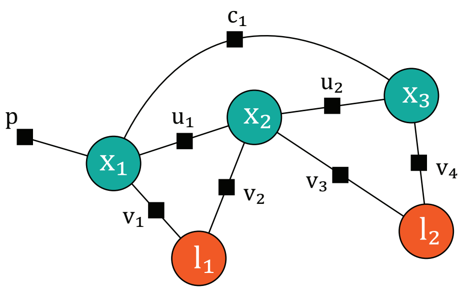
Figure 6.2 , were included. As the uniform weighting heuristic of Equation 6.5 aims to follow these minima as closely as possible, it is as expected that these two lines are close together. The non-uniform weighting heuristic, however, out-performed these weights. The heuristic of Equation 6.4, from TRW-NBP, was also included for comparison. This performed slightly worse than our weights and resulted increased RMSE for low connectivities on the $N = 10$ graph.

6.6 Application to factor graph localisation

As a practical example of this algorithm, our weights were tested on a linear multi-robot *simultaneous localisation and mapping* (SLAM) problem. SLAM uses measurements between robot poses and landmarks to localise the robots and map the landmarks. Graph-SLAM represents this as *maximum a posteriori* optimisation on a factor graph [5]. BP has been shown to efficiently solve graphical representations in localisation [273], tracking [279], and SLAM [280]. The difference between an MRF, as discussed thus far in the chapter, and factor graph can be seen in Figure 6.6a, with visual representation of graph SLAM shown in Figure 6.6b. Nodes are the same as those on an MRF, whilst the constraints between nodes and priors are



(a) MRFs and factor graphs



(b) Factor graph representation of the SLAM problem.

Figure 6.6: Factor graphs, (a) the conversion from an MRF with constraints between nodes u and prior information p as factors, and (b) SLAM as a factor graph with x a pose, l a landmark, p the prior, u a constraint between poses, v a measurement of a landmark, and c a loop-closure.

now shown as factors represented by squares. The factor graph representation given in Figure 6.6b is a simplified version of that in Figure 2.23 discussed in Section 2.4.2.

The simulation of our weighting method on the linear multi-robot SLAM problem was done with a variable number of robots moving randomly on a 100m$\times$100m plane with four landmarks. To keep the problem linear for demonstration purposes, we are only considering $(x,\ y)$ positions. At each iteration, the robots measure the distance to its previous pose and any landmarks or robots in range. The measurement radius was 30m and were affected by zero-mean additive Gaussian noise of variance 3m. This simulation was repeated for different numbers of robots ranging from 1 to 6, for 1000 trials each, to measure the average reduction in RMSE compared to unweighted GaBP.

Figure 6.7 shows an output of this simulation performed with four robots, with the ground truth paths shown as solid lines. The unweighted GaBP solution is shown as the dashed line, whilst the non-uniform weighted GaBP is shown as the dotted line, which was closer to the ground truth. This demonstrates a small but clear improvement in the localisation accuracy for our linearised scenario.

The RMSE reduction for 1000 trials of 1 to 6 robots is shown in Figure 6.8. As the number of robots increases, and therefore the graph size, the effect of the weighted messages on RMSE increases as well. The TRW weights, however, decrease in effectiveness as the graph size increases. This is understandable as the weights being used in our methods are designed specifically for the type of network seen in this robotics example.

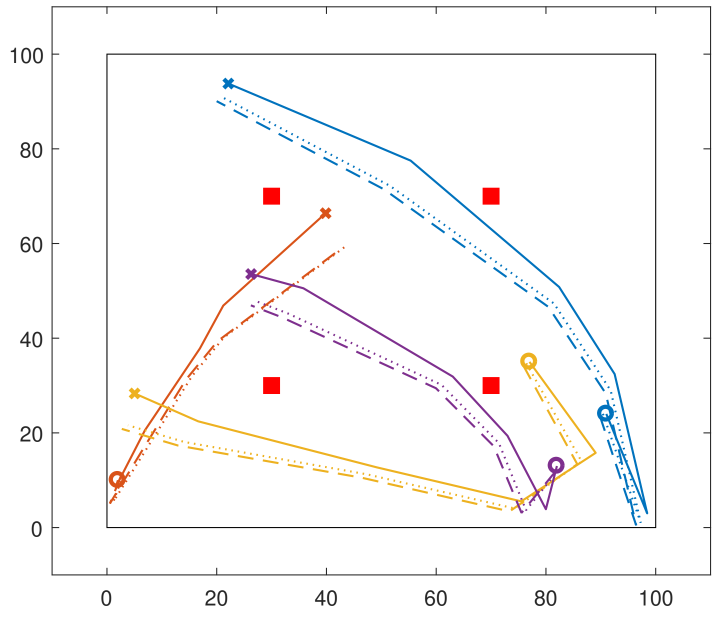


Figure 6.7: Example result of using weighted GaBP to solve a 2D graph-SLAM problem.

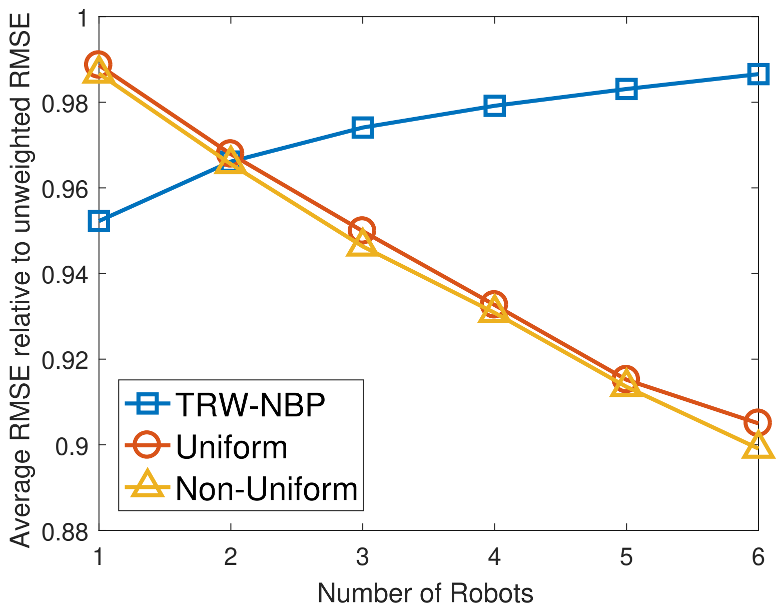


Figure 6.8: Relative reduction in RMSE compared to unweighted GaBP for different numbers of robots, using TRW weights as well as our uniform and non-uniform weights.

Table 6.1: Comparison of the GaBP procedure used in Chapter 3 to the uniform and non-uniform weighted GaBP, evaluated for average iterations per GaBP alignment and RMSE of the resulting calibration.

Weighting	Iterations	RMSE
Unweighted	27.4	0.486px
Uniform	38.8	0.461px
Non-uniform	19.1	0.453px

Table 6.2: Comparison of the GaBP procedure used in Chapter 4 to the uniform and non-uniform weighted GaBP, evaluated for average iterations per GaBP alignment and RMSE of the resulting calibration.

Weighting	Iterations	RMSE
Unweighted	12.8	0.171px
Uniform	17.6	0.163px
Non-uniform	8.9	0.161px

6.7 Application to distributed calibration

In this dissertation, two forms of distributed one-dimensional calibration and localisation were presented. In Chapter 3, a distributed version of Zhang's original one-dimensional calibration algorithm was presented [15]. In Chapter 4, an improved version of the multi-view general-motion one-dimensional calibration algorithm was utilised [87]. Both of these Chapters utilised GaBP as the core of their consensus algorithms and operated on graphs of a similar size to those discussed in this chapter; Chapter 3 artificially constructed a lattice graph of ten nodes while Chapter 4 generated a small-world network of ten nodes from common observations. The main goal of this chapter is to present a method of reducing RMSE in GaBP on cyclic graphs like those found in these two networks, therefore, the effect of the edge weights on those two algorithms is presented here.

For the first calibration method of Chapter 3, the GaBP procedure described in Section 3.6 was replaced by the weighted GaBP of this chapter and evaluated on the real-image dataset of Section 3.7.3. This was evaluated to compare the original GaBP implementation to both the uniform and non-uniform weights in terms of GaBP iterations per optimisation iteration as well as RMSE of the resulting calibration, as reported in Table 6.1. Similarly for the second calibration method of Chapter 4, the GaBP procedure of Section 4.5.2 was replaced with the improved GaBP and evaluated on the real-image dataset of Section 4.6.2. The results evaluating the GaBP iterations per outer iteration and RMSE for the original GaBP, uniformly-weighted GaBP, and non-uniformly-weighted GaBP for this second calibration algorithm are reported in Table 6.2.

As can be seen, both the uniform and non-uniform variants of the algorithm produce similar reduction in RMSE, with the non-uniform weights performing very slightly better. Both algorithms utilised graphs that had moderate connectivities; in Chapter 3, the graphs were lattice graph where the connectivity was intentionally limited to avoid the higher errors seen in heavily cyclical graphs, and similarly, in Chapter 4, the graph connections were limited by co-visibility of the calibration object. Therefore, the reduction in RMSE being in similar magnitude to the 10 and 20 node simulated graphs is expected. An interesting outcome, however, is that the uniform weights increase the average number of iterations required for convergence but about 40% whereas the non-uniform weights decrease the average number of iterations by almost one third. This reinforces that the non-uniform weighting scheme is preferable choice.

6.8 Summary

This chapter has presented two weighting systems to reduce RMSE in Gaussian belief propagation on cyclic graphs, with the goal of improving the accuracy of the distributed algorithms being utilised in the multi-robot pipeline discussed throughout this dissertation. The first method involves uniform weights based on graph size and average node degree, whilst the second method uses non-uniform weights from the average degree at the nodes on either end of an edge. These two weighting systems were empirically derived by analysing the RMSE minima for uniform weights on small-world graphs of varying sizes and connectivities.

The results have shown that these weights effectively reduce the error in the GaBP solution, with more significant effects on graphs of higher connectivity. It was shown that in small-world networks, the proposed weighting method performed better than the tree-reweighted alternative. Furthermore, the non-uniform weighting method outperformed the uniform weighting method. These results were shown both in an example application of linearised multi-robot graph SLAM as well as in the distributed calibration algorithms discussed throughout this dissertation, with the non-uniform weights achieving a more accurate solution in fewer iterations compared to the unweighted GaBP. We will be continuing this work to better relate the optimal message weights to the intrinsic properties of the graph and will also investigate applying it to further robotics problems.

Chapter 7

Conclusions and future work

7.1 Overview

Advancements in robotic networks with vision applications has led to the need for robust distributed algorithms that allow for greater robustness and scalability of the systems [1, 2]. Standard centralised solutions have many issues with singular points of failure, communication bottlenecks, and computational inefficiencies. Distributed alternatives reduce the problem to a combination of local computations and information sharing with direct neighbours which removes the single point of failure, reduces the communication load, and lowers the computational requirement for any one processor [3]. With these favourable characteristics, distributed algorithms have seen wide application in robotic vision networks for addressing the problems of calibration, localisation, visual odometry, and SLAM. This has raised questions as to how these tasks can be formulated in an entirely distributed manner with equivalent performance to centralised alternatives, how these algorithms can be designed to operate effectively within the constraints of low-cost on-board processing, and how they can be incorporated into a consistent pipeline. This work contributed a range of algorithms to this class of problems in three key stages of the pipeline: initialisation, local tracking, and global mapping. The goal of these algorithms was to improve capabilities for robotic vision platforms to perform calibration, localisation, and mapping.

Calibration and localisation are the crucial first steps for a vision network to obtain an initialisation; the calibration provides knowledge of the intrinsic parameters of the camera model and the localisation provides relative pose information between the vision nodes [4]. This allows three-dimensional metric information to be determined from the scene, as seen by the robots, and produces an accurate starting point for the application of higher-order robotic vision tasks such as mapping. Distributed approaches to this problem have mostly focused on the use of self-calibration algorithms [7], or exclusively focused on the localisation sub-problem [8], and have solved the problem by utilising distributed algorithms such as belief propagation and gradient descent through average consensus.

Following the initial calibration and localisation, a robotic vision network can perform tasks such as visual odometry and SLAM [5]. This can be seen as an extension to the localisation problem, where the robots track their motion through an unknown scene whilst producing a map. Many state-of-the-art visual SLAM implementations can be interpreted using a keyframe-SLAM framework, where information from the images, tracking, and mapping procedures are collected into certain images that are kept as 'keyframes', on which optimisation is performed. Furthermore, most visual SLAM algorithms fall into the paradigm distinction between indirect methods [20, 100], which use robust feature descriptors, and direct methods [92, 93], which operate directly on the raw intensity values. Most distributed multi-robot SLAM implementations rely on indirect methods due to the ability to use robust feature descriptors for wide-baseline matching. Distributed SLAM systems generally focus on applying distributed algorithms following a single-robot visual odometry in order to perform map alignment [6, 24], merging [236, 237], and global optimisation processes [25, 27, 238].

Following the thorough exploration of state-of-the-art calibration, localisation, visual odometry and SLAM algorithms, as well as their application to distributed multi-robot systems, presented in Chapter 2, a clear three-stage pipeline was identified. Firstly, the system requires an initialisation which is provided by the calibration and localisation. This requires communication between nodes and distributed processing to ensure that a globally consistent state is determined. Following this, a standard single-robot visual odometry is performed to track the motion of each robot independently and produce local maps. Finally, inter-robot communication and distributed algorithms are again leveraged to perform global mapping that combines and optimises the trajectories and maps of all robots together. It is to these three stages of the pipeline that this dissertation has contributed algorithms and furthered understanding, underpinning the three primary goals.

7.2 Research summary

The three goals that were explored in this dissertation were as follows: the development of a robust calibration and localisation algorithm that can initialise a distributed multi-robot platform to an accurate metric state of known scale, as was explored in Chapters 3 and 4; the development of a single-robot visual odometry method that is suitable for the second stage, with a focus on computational efficiency and data-type homogeneity, as was explored in Chapter 5; and finally, the improvement of convergence and accuracy of distributed algorithms for use in the third stage, as was explored in Chapter 6.

In Chapter 3, a method for initialising a distributed robotic vision network was explored based on the one-dimensional camera calibration algorithm of Zhang [15] and the Gaussian belief propagation proposed by Weiss et al. [200]. Existing distributed camera sensor network calibration algorithms are based on self-calibration, such as in the work by Devarajan et al. [7] or Tron and Vidal [8], which cannot determine the scale of the scene. The introduction of a calibration object resolves this ambiguity, however, not all calibration objects are made equal; one-dimensional calibration objects were found to be ideal for the distributed camera system due to the lack of self-occlusion at wide angles. Furthermore, the introduction of a calibration object simplified the construction of the vision graph and initial estimation of extrinsic parameters through the alignment of the 3D points associated with the object observations, providing known correspondences. Following this, a cluster-based bundle adjustment was performed at each node on the lattice graph, which were all brought into global consensus using Gaussian belief propagation. Evaluations of this method demonstrated that the distributed algorithm performed to similar accuracy to an equivalent centralised global bundle adjustment, showing that the process could be effectively distributed across the network without significant losses in accuracy.

Chapter 4 expanded upon the method of Chapter 3 by improving it in a number of ways. Firstly, the single-camera calibration method of Zhang [15] was replaced by the multi-camera one-dimension calibration and localisation method of de França [87] which couples the intrinsic and extrinsic calibration together. This method was improved upon by applying better numerical condition through normalisation of the projective reconstruction as well as condensing the multiple stages of nonlinear optimisation into a single stage. As with Chapter 3, the presence of the calibration object was then leveraged to initialise the vision graph, however, in this algorithm small-world network structures were utilised to ensure better convergence of the distributed algorithm. Finally, the distributed consensus of the previous method, which was essentially a variance-weighted distributed average of local objectives, was overhauled to ensure that the global objective was directly equivalent to a centralised global bundle adjustment through the use of the alternating direction method of multipliers [215]. This used an augmented Lagrangian approach to split the global objective into local objectives with included consensus costs that were brought into alignment using Gaussian belief propagation distributed averaging. Extensive testing on real and simulated data demonstrated that the centralised version of the algorithm showed superior performance to the original algorithm, self-calibration, and 2D calibration for the extrinsic and a number of intrinsic parameters, with the distributed algorithm achieving similar accuracy to the centralised algorithm. This reinforced the findings of the previous chapter — that distributed methods could effectively split global objectives without losses in accuracy. Overall, these two methods provided a highly accurate initialisation stage to a distributed robotic vision network that was able to provide the localisation to a known scale.

The second stage of the pipeline was explored in Chapter 5 with the goal of developing a computationally efficient visual odometry method that is suitable for the low-powered hardware found on many multi-robot platforms whilst maintaining the use of the robust feature descriptors necessary for wide-baseline matching between robots. The direct methods paradigm for visual odometry was identified as a significant trend in the field that produces highly efficient algorithms compared to indirect methods [100, 107]. Despite this, direct methods still ultimately compute robust feature descriptors for use in wide-baseline matching [107]. Therefore, a hybrid approach was taken to incorporate robust binary feature descriptors into the direct whole-image alignment process in order to maintain the efficiency of direct methods whilst improving data-type homogeneity. This method was developed based on consideration of the Census transform [260], which produces primitive binary descriptors. Although these are matched using the Hamming norm, which cannot be in gradient descent, the related Rank transform is matched using the standard L_2 norm and can be used in gradient descent. From this, an approach was developed where residuals were compared using the Hamming norm but gradients and descent direction were determined by comparison of Hamming weights. This was expanded to binary descriptors of arbitrary size and complexity and tested for a number of widely used binary descriptors on real data. Results showed that this was significantly more accurate than direct tracking on intensity and, when the descriptors were only computed sparsely, computational requirements were not increased too greatly. When considering accuracy alone, ORB descriptors were found to be the best choice amongst the types considered [101], however, if considering both accuracy and efficiency, then BRIEF16 descriptors were found to be best [152]. This method provided a robust and computationally efficient visual odometry algorithm that was suitable for the second stage in the distributed robotic vision pipeline being considered, as its use of robust descriptors lends well to incorporation into the global mapping of the third stage.

Chapter 6 contributed to the accuracy and efficiency of the algorithms used in this third stage, as well as in the first stage, through improvements to convergence properties of Gaussian belief propagation [200, 242] for small-world networks [278]. Existing methods for accelerating the convergence for belief propagation, and therefore improving accuracy, use edge-weights which are applied to the messages [211, 276, 277]. These have been computed based on the probability that a given edge appears in the set of all spanning-trees for the cyclic graph, however, computation of all spanning-trees is intractable for even moderately sized graphs. Alternatives have estimated these probabilities using empirical approaches based on global information about the network structure [211, 277]. The method proposed in Chapter 6 derived an empirical approach that uses the node degrees at either end of a given edge, reducing the amount of global information required at each node. The precise weighting values were determined through the analysis of small-world networks of a range of sizes. Results showed that the reduction in RMSE compared to the unweighted alternative became more significant as the size of the graph increased. The method was also tested on a simple linearised multi-robot graph-SLAM problem, showing clear improvement in the localisation accuracy and again increasing the level of improvement as the size of the graph increased. This demonstrated that the weighting scheme was suitable for improving convergence speed and accuracy of Gaussian belief propagation and therefore was beneficial for use in the distributed algorithms for stages one and three of the pipeline.

Overall, the contributions of this dissertation can be summarised as three-fold: the development of algorithms for the robust initialisation of a distributed multi-robot vision platform, the development of a visual odometry approach that is computationally efficient for use in low-powered hardware whilst utilising suitable data-types for inter-robot matching in distributed networks, and finally, the improvement of consensus speed and accuracy for the underlying distributed algorithm that help align the local estimates into a global consensus. Through this, the three questions have been addressed: algorithms have been presented for computing global objectives in entirely distributed manners, algorithms have been proposed to meet the needs of low-cost on-board processing, and finally, these algorithms have been incorporated into a consistent pipeline for distributed robotic vision platforms.

7.3 Future work

The work presented in this dissertation has opened up a plethora of avenues for future work investigating this distributed robotic vision pipeline for calibration, localisation. and mapping.

- The work presented in Chapter 4 developed upon the work of Devarajan et al. [7] by replacing the Gaussian belief propagation distributed averaging with an ADMM-based optimisation procedure [215]. In Chapter 4, this was only applied to one-dimensional calibration, however, it is equally suitable for application to self-calibration [16, 17, 47]. Although self-calibration cannot determine the scale of the scene, it has less strict requirements on all cameras observing a calibration object. Application of the ADDM bundle adjustment to the self-calibration approach of Devarajan et al. [7] could improve the accuracy whilst producing a method with better ease-of-use in situations where known scene scale is not needed.

- Although ADMM provides a straight-forward method for producing a distributed algorithm from separable problems, it is not always the best choice for doing this and is best used for when the local objective updates can be computed analytically [215]. In the application presented in Chapter 4, the algorithm ultimately performs an optimisation problem within the iterations of an optimisation problem. The impact of this on computation time is minimised by only performing one iteration of the inner problem for each outer iteration, however, alternative approaches can be explored. For example, the average consensus-based gradient descent method of Tron and Vidal [8] could be extended to the full calibration problem, including intrinsic parameters, to replace the ADMM method entirely. On the other hand, the relaxation used by Choudhary et al. [27] could be used within the ADMM approach to improve the accuracy of the inner optimisation compared to simply performing one iteration of gradient descent, therefore reducing the number of outer iterations required.

- Continuing the consideration of distributed bundle adjustment, the ADMM approach could be extended to the type of bundle adjustment used in the local and global optimisations in SLAM, as well as altered to the form used in pose-graph optimisation. This is similar to the distributed bundle adjustment proposed by Eriksson et al. [223] who used proximal splitting methods, however, their method was split in two different ways: firstly, by disjoint partitions of images, and secondly, by the image points. This makes their algorithm excellent for parallel processing, but becomes more complicated on a distributed robotic network where processing is ideally only separated by robots. The general consensus form of ADMM could produce a version of their algorithm that is more clearly separated by robots for use in stage three of the pipeline discussed in this dissertation.

- Further work is being done on the method proposed in Chapter 5 for a hybrid feature-based direct visual odometry. To compare the method more directly to state-of-the-art visual odometry methods, such as DSO [93], the algorithm should be incorporated into the keyframe-based pipeline discussed in Section 2.4.2. This would allow decisions over sparse point selection as well as keyframe selection and management to be designed around both the usage of specific robust feature descriptors as well as the direct alignment procedure. Furthermore, analysis of the performance of each descriptor type in wide-baseline matching would better inform descriptor choice when considering integration into a multi-robot system.

- Considering an alternative approach to the single-robot visual odometry discussed in Chapter 5, rather than incorporating feature descriptors into the direct alignment process, the results of the direct alignment process could be incorporated into the global mapping step instead. Currently, most direct methods rely on robust feature matching for wide-baseline pose estimation and loop closure [107], which was the motivation for developing the algorithm in Chapter 5. However, feature descriptors could be avoided entirely if the map generated by direct methods could be used for loop closure. The inverse depth-map representation of world points used in direct methods is similar to the point cloud representation used in RGB-D SLAM methods, such as KinectFusion [90] and ElasticFusion [96], which have been shown to perform accurate loop closure on portions of this point cloud. Therefore, a similar approach could be explored in direct methods to avoid feature descriptors entirely.

- Finally, in Chapter 6 an empirical method for choosing edge-weights was applied to small-world networks of a range of sizes based on node degrees. This should be explored in greater detail to relate these weights more directly to the properties of the graph. This is also related to the attenuation factor discussed in the work of Moallemi and Van Roy [201] which was used to accelerate convergence, however, like with the edge appearance potentials of Wainwright et al. [276], Wymeersch et al. [277], and Savic et al. [211], decisions on the optimal value for the attenuation factor was based on global information. Therefore, research into appropriate edge-weights and attenuation factors based entirely on local information would be highly useful, particularly on time-changing graphs where these values could change.

Bibliography

[1] B. Song, C. Ding, A. T. Kamal, J. A. Farrell and A. K. Roy-chowdhury, 'Distributed camera networks,' *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 20–31, 2011. DOI: `10.1109/MSP.2011.940441`.

[2] R. Tron and R. Vidal, 'Distributed computer vision algorithms through distributed averaging,' in *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, USA: IEEE Computer Society, 2011, pp. 57–63. DOI: `10.1109/CVPR.2011.5995654`.

[3] R. J. Radke, 'A survey of distributed computer vision algorithms,' in *Handbook of Ambient Intelligence and Smart Environments*, H. Nakashima, H. Aghajan and J. C. Augusto, Eds. Boston, MA: Springer US, 2010, pp. 35–55, ISBN: 978-0-387-93808-0. DOI: `10.1007/978-0-387-93808-0_2`.

[4] Z. Zhang, 'Camera calibration,' in *Emerging Topics in Computer Vision*, G. Medioni and S. B. Kang, Eds., Prentice Hall PTR, 2005, ch. 2, pp. 5–43, ISBN: 978-0-13-101366-7.

[5] C. Cadena *et al.*, 'Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,' *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016. DOI: `10.1109/TRO.2016.2624754`.

[6] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz and B. Stewart, 'Distributed multirobot exploration and mapping,' *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006. DOI: `10.1109/JPROC.2006.876927`.

[7] D. Devarajan, Z. Cheng and R. J. Radke, 'Calibrating distributed camera networks,' *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1625–1639, 2008. DOI: `10.1109/JPROC.2008.928759`.

[8] R. Tron and R. Vidal, 'Distributed 3-d localization of camera sensor networks from 2-d image measurements,' *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3325–3340, 2014. DOI: `10.1109/TAC.2014.2351912`.

[9] A. Cunningham, V. Indelman and F. Dellaert, 'Ddf-sam 2.0: Consistent distributed smoothing and mapping,' in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 5220–5227. DOI: `10.1109/ICRA.2013.6631323`.

[10] D. C. Brown, 'Close-range camera calibration,' *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, 1971.

[11] J. Heikkila and O. Silven, 'A four-step camera calibration procedure with implicit image correction,' in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 1997, pp. 1106–1112. DOI: `10.1109/CVPR.1997.609468`.

[12] R. Tsai, 'A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses,' *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987. DOI: `10.1109/JRA.1987.1087109`.

[13] Z. Zhang, 'A flexible new technique for camera calibration,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. DOI: `10.1109/34.888718`.

[14] ——, 'Camera calibration with one-dimensional objects,' in *Proceedings of European Conference on Computer Vision*, A. Heyden, G. Sparr, M. Nielsen and P. Johansen, Eds., Springer, Berlin, Heidelberg, 2002, pp. 161–174, ISBN: 978-3-540-47979-6. DOI: `10.1007/3-540-47979-1_11`.

[15] ——, 'Camera calibration with one-dimensional objects,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, pp. 892–899, 2004. DOI: `10.1109/TPAMI.2004.21`.

[16] O. D. Faugeras, Q.-T. Luong and S. J. Maybank, 'Camera self-calibration: Theory and experiments,' in *Proceedings of European Conference on Computer Vision*, G. Sandini, Ed., Springer, Berlin, Heidelberg, 1992, pp. 321–334, ISBN: 978-3-540-47069-4. DOI: `10.1007/3-540-55426-2_37`.

[17] Q.-T. Luong and O. D. Faugeras, 'Self-calibration of a moving camera from point correspondences and fundamental matrices,' *International Journal of Computer Vision*, vol. 22, no. 3, pp. 261–289, 1997. DOI: `10.1023/A:1007982716991`.

[18] M. Pollefeys and L. Van Gool, 'A stratified approach to metric self-calibration,' in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 1997, pp. 407–412. DOI: `10.1109/CVPR.1997.609357`.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, ser. Cambridge Books Online. Cambridge University Press, 2003, ISBN: 978-0-521-54051-3.

[20] G. Klein and D. Murray, 'Parallel tracking and mapping for small ar workspaces,' in *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE, 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[21] M. Cummins and P. Newman, 'Appearance-only slam at large scale with fab-map 2.0,' *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011. DOI: 10.1177/0278364910385483.

[22] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford and G. Wyeth, 'Openfabmap: An open source toolbox for appearance-based loop closure detection,' in *Proceedings of the International Conference on Robotics and Automation*, St Paul, Minnesota: IEEE, 2011. DOI: 10.1109/ICRA.2012.6224843.

[23] D. Gálvez-López and J. D. Tardos, 'Bags of binary words for fast place recognition in image sequences,' *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012. DOI: 10.1109/TRO.2012.2197158.

[24] X. S. Zhou and S. I. Roumeliotis, 'Multi-robot slam with unknown initial correspondence: The robot rendezvous case,' in *Proceedings of 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 1785–1792. DOI: 10.1109/IROS.2006.282219.

[25] J. Knuth and P. Barooah, 'Collaborative 3d localization of robots from relative pose measurements using gradient descent on manifolds,' in *Proceedings of 2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 1101–1106. DOI: 10.1109/ICRA.2012.6225066.

[26] ——, 'Collaborative localization with heterogeneous inter-robot measurements by riemannian optimization,' in *Proceedings of 2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 1534–1539. DOI: 10.1109/ICRA.2013.6630774.

[27] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen and F. Dellaert, 'Distributed trajectory estimation with privacy and communication constraints: A two-stage distributed gauss-seidel approach,' in *Proceedings of 2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 5261–5268. DOI: 10.1109/ICRA.2016.7487736.

[28] A. Cunningham, M. Paluri and F. Dellaert, 'Ddf-sam: Fully distributed slam using constrained factor graphs,' in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3025–3030. DOI: 10.1109/IROS.2010.5652875.

[29] R. Olfati-Saber, 'Flocking for multi-agent dynamic systems: Algorithms and theory,' *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006. DOI: 10.1109/TAC.2005.864190.

[30] R. Tron, J. Thomas, G. Loianno, K. Daniilidis and V. Kumar, 'A distributed optimization framework for localization and formation control: Applications to vision-based measurements,' *IEEE Control Systems Magazine*, vol. 36, no. 4, pp. 22–44, 2016. DOI: 10.1109/MCS.2016.2558401.

[31] L. Xiao, S. Boyd and S. Lall, 'A scheme for robust distributed sensor fusion based on average consensus,' in *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks*, IEEE, 2005, pp. 63–70. DOI: 10.1109/IPSN.2005.1440896.

[32] M. Taj and A. Cavallaro, 'Distributed and decentralized multicamera tracking,' *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 46–58, 2011. DOI: 10.1109/MSP.2011.940281.

[33] R. Tron and R. Vidal, 'Distributed image-based 3-d localization of camera sensor networks,' in *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, IEEE, 2009, pp. 901–908. DOI: 10.1109/CDC.2009.5400405.

[34] R. Olfati-Saber and R. M. Murray, 'Consensus problems in networks of agents with switching topology and time-delays,' *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004. DOI: 10.1109/TAC.2004.834113.

[35] R. Olfati-Saber, J. A. Fax and R. M. Murray, 'Consensus and cooperation in networked multi-agent systems,' *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007. DOI: 10.1109/JPROC.2006.887293.

[36] B. Halloran, P. Premaratne, P. Vial and I. Kadhim, 'Distributed one dimensional calibration and localisation of a camera sensor network,' in *Proceedings of ICIC 2017: Intelligent Computing Theories and Application*, D.-S. Huang, K.-H. Jo and J. C. Figueroa-García, Eds., Springer International Publishing, 2017, pp. 581–593, ISBN: 978-3-319-63312-1. DOI: 10.1007/978-3-319-63312-1_51.

[37] B. Halloran, P. Premaratne and P. J. Vial, 'Optimizing edge weights for distributed inference with gaussian belief propagation,' in *Proceedings of ICIC 2018: Intelligent Computing Theories and Application*, D.-S. Huang, V. Bevilacqua, P. Premaratne and P. Gupta, Eds., Springer International Publishing, 2018, pp. 46–59, ISBN: 978-3-319-95930-6. DOI: `10.1007/978-3-319-95930-6_6`.

[38] B. Halloran, P. Premaratne, P. J. Vial and I. Kadhim, 'Single and multi-channel direct visual odometry with binary descriptors,' in *Proceedings of ICIC 2019: Intelligent Computing Methodologies*, D.-S. Huang, Z.-K. Huang and A. Hussain, Eds., Springer International Publishing, 2019, pp. 86–98, ISBN: 978-3-030-26766-7. DOI: `10.1007/978-3-030-26766-7_9`.

[39] B. Halloran, P. Premaratne and P. J. Vial, 'Robust one-dimensional calibration and localisation of a distributed camera sensor network,' *Pattern Recognition*, vol. 98, 107058, pp. 1–12, 2020. DOI: `10.1016/j.patcog.2019.107058`.

[40] I. J. Kadhim, P. Premaratne, P. J. Vial and B. Halloran, 'A comparative analysis among dual tree complex wavelet and other wavelet transforms based on image compression,' in *Proceedings of ICIC 2017: Intelligent Computing Theories and Application*, D.-S. Huang, K.-H. Jo and J. C. Figueroa-García, Eds., Springer International Publishing, 2017, pp. 569–580, ISBN: 978-3-319-63312-1. DOI: `10.1007/978-3-319-63312-1_50`.

[41] Q. Al-Shebani, P. Premaratne, P. J. Vial, D. J. McAndrew and B. Halloran, 'Co-simulation method for hardware/software evaluation using xilinx system generator: A case study on image compression algorithms for capsule endoscopy,' in *Proceedings of 2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, IEEE, 2018, pp. 1–4. DOI: `10.1109/ICSPCS.2018.8631737`.

[42] C. Shiranthika, P. Premaratne, Z. Zheng and B. Halloran, 'Realtime computer vision-based accurate vehicle counting and speed estimation for highways,' in *Proceedings of ICIC 2019: Intelligent Computing Methodologies*, D.-S. Huang, Z.-K. Huang and A. Hussain, Eds., Springer International Publishing, 2019, pp. 583–592, ISBN: 978-3-030-26766-7. DOI: `10.1007/978-3-030-26763-6_56`.

[43] I. J. Kadhim, P. Premaratne, P. J. Vial and B. Halloran, 'Comprehensive survey of image steganography: Techniques, evaluations, and trends in future research,' *Neurocomputing*, vol. 335, pp. 299–326, 2019, ISSN: 0925-2312. DOI: `10.1016/j.neucom.2018.06.075`.

[44] P. Premaratne, M. Abdullah, I. Kadhim, B. Halloran and P. Vial, 'Optimization of low-speed dual rotor axial flux generator design through electromagnetic modelling and simulation,' in *Proceedings of ICIC 2021: Intelligent Computing Theories and Application*, D.-S. Huang, K.-H. Jo, J. Li, V. Gribova and V. Bevilacqua, Eds., Springer International Publishing, 2021, pp. 786–801, ISBN: 978-3-030-84522-3. DOI: `10.1007/978-3-030-84522-3_64`.

[45] R. I. Hartley, 'In defense of the eight-point algorithm,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997. DOI: `10.1109/34.601246`.

[46] ——, 'Estimation of relative camera positions for uncalibrated cameras,' in *Proceedings of European Conference on Computer Vision*, Berlin, Heidelberg: Springer, 1992, pp. 579–587, ISBN: 978-3-540-47069-4. DOI: `10.1007/3-540-55426-2_62`.

[47] B. Triggs, 'Autocalibration and the absolute quadric,' in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 1997, pp. 609–614. DOI: `10.1109/CVPR.1997.609388`.

[48] R. I. Hartley, 'Self-calibration from multiple views with a rotating camera,' in *Proceedings of European Conference on Computer Vision*, Berlin, Heidelberg: Springer, 1994, pp. 471–478, ISBN: 978-3-540-48398-4. DOI: `10.1007/3-540-57956-7_52`.

[49] M. Armstrong, A. Zisserman and R. Hartley, 'Self-calibration from image triplets,' in *Proceedings of European Conference on Computer Vision*, Springer, Berlin, Heidelberg: Springer, 1996, pp. 1–16, ISBN: 978-3-540-49949-7. DOI: `10.1007/BFb0015519`.

[50] P. Sturm, 'Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction,' in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 1997, pp. 1100–1105. DOI: `10.1109/CVPR.1997.609467`.

[51] ——, 'A case against kruppa's equations for camera self-calibration,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1199–1204, 2000. DOI: `10.1109/34.879804`.

[52] Y. Gao and H. Radha, 'A multistage camera self-calibration algorithm,' in *Proceedings of 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, IEEE, vol. 3, 2004, pp. iii–537. DOI: `10.1109/ICASSP.2004.1326600`.

[53] T. Köhler, ‘Comparison of self-calibration algorithms for moving cameras,’ M.S. thesis, University of Applied Sciences Munich, 2010.

[54] B. Zhuang, Q.-H. Tran, G. H. Lee, L. F. Cheong and M. Chandraker, ‘Degeneracy in self-calibration revisited and a deep learning solution for uncalibrated slam,’ in *Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 3766–3773. DOI: `10.1109/IROS40897.2019.8967912`.

[55] C. Zhang, F. Rameau, J. Kim, D. M. Argaw, J.-C. Bazin and I. S. Kweon, ‘Deepptz: Deep self-calibration for ptz cameras,’ in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 1041–1049.

[56] Z. Tang, Y.-S. Lin, K.-H. Lee, J.-N. Hwang and J.-H. Chuang, ‘Esther: Joint camera self-calibration and automatic radial distortion correction from tracking of walking humans,’ *IEEE Access*, vol. 7, pp. 10 754–10 766, 2019. DOI: `10.1109/ACCESS.2019.2891224`.

[57] Y. Abdel-Aziz and H. Karara, ‘Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry,’ in *Proceedings of the Symposium on Close-Range Photogrammetry*, 1971, pp. 1–18. DOI: `10.14358/PERS.81.2.103`.

[58] J. Weng, P. Cohen, M. Herniou *et al.*, ‘Camera calibration with distortion models and accuracy evaluation,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992. DOI: `10.1109/34.159901`.

[59] Z. Zhang, ‘Determining the epipolar geometry and its uncertainty: A review,’ *International Journal of Computer Vision*, vol. 27, no. 2, pp. 161–195, 1998. DOI: `10.1023/A:1007941100561`.

[60] P. F. Sturm and S. J. Maybank, ‘On plane-based camera calibration: A general algorithm, singularities, applications,’ in *Proceedings of 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, IEEE, vol. 1, 1999, pp. 432–437. DOI: `10.1109/CVPR.1999.786974`.

[61] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O’Reilly Media, 2008, ISBN: 978-0-596-55404-0.

[62] J.-Y. Bouguet. ‘Camera calibration toolbox for matlab.’ (2004), [Online]. Available: `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html` (visited on 30/04/2021).

[63] S. Daftry, M. Maurer, A. Wendel and H. Bischof, ‘Flexible and user-centric camera calibration using planar fiducial markers.,’ in *Proceedings of 2013 British Machine Vision Conference*, T. Burghardt, D. Damen, W. Mayol-Cuevas and M. Mirmehdi, Eds., BMVA Press, 2013, 19, pp. 1–13, ISBN: 1-901725-49-9. DOI: 10.5244/C.27.19.

[64] C. Schmalz, F. Forster and E. Angelopoulou, ‘Camera calibration: Active versus passive targets,’ *Optical Engineering*, vol. 50, no. 11, 113601, pp. 1–11, 2011. DOI: 10.1117/1.3643726.

[65] M. Fiala, ‘Designing highly reliable fiducial markers,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1317–1324, 2009. DOI: 10.1109/TPAMI.2009.146.

[66] M. Fiala and C. Shu, ‘Self-identifying patterns for plane-based camera calibration,’ *Machine Vision and Applications*, vol. 19, no. 4, pp. 209–216, 2008. DOI: 10.1007/s00138-007-0093-z.

[67] B. Atcheson, F. Heide and W. Heidrich, ‘Caltag: High precision fiducial markers for camera calibration.,’ in *Proceedings of the 15th International Workshop on Vision, Modeling and Visualization 2010 (VMV 2010)*, R. Koch, A. Kolb and C. Rezk-salama, Eds., vol. 10, The Eurographics Association, 2010, pp. 41–48, ISBN: 978-3-905673-79-1. DOI: 10.2312/PE/VMV/VMV10/041-048.

[68] P. Hammarstedt, P. Sturm and A. Heyden, ‘Degenerate cases and closed-form solutions for camera calibration with one-dimensional objects,’ in *Proceedings of Tenth IEEE International Conference on Computer Vision (ICCV’05)*, IEEE, vol. 1, 2005, pp. 317–324. DOI: 10.1109/ICCV.2005.68.

[69] F. Wu, Z. Hu and H. Zhu, ‘Camera calibration with moving one-dimensional objects,’ *Pattern Recognition*, vol. 38, no. 5, pp. 755–765, 2005, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2004.11.005.

[70] F. Qi, Q. Li, Y. Luo and D. Hu, ‘Camera calibration with one-dimensional objects moving under gravity,’ *Pattern Recognition*, vol. 40, no. 1, pp. 343–345, 2007, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2006.06.029.

[71] ——, ‘Constraints on general motions for camera calibration with one-dimensional objects,’ *Pattern Recognition*, vol. 40, no. 6, pp. 1785–1792, 2007, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2006.11.001.

[72] J. A. de França, M. R. Stemmer, M. B. d. M. França and E. G. Alves, ‘Revisiting zhang’s 1d calibration algorithm,’ *Pattern Recognition*, vol. 43, no. 3, pp. 1180–1187, 2010, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2009.08.001.

[73] K. Shi, Q. Dong and F. Wu, 'Weighted similarity-invariant linear algorithm for camera calibration with rotating 1-d objects,' *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3806–3812, 2012. DOI: `10.1109/TIP.2012.2195013`.

[74] L. Wang, F. Duan and K. Lu, 'An adaptively weighted algorithm for camera calibration with 1d objects,' *Neurocomputing*, vol. 149, pp. 1552–1559, 2015, ISSN: 0925-2312. DOI: `10.1016/j.neucom.2014.08.037`.

[75] M. Pollefeys, R. Koch and L. Van Gool, 'Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters,' *International Journal of Computer Vision*, vol. 32, no. 1, pp. 7–25, 1999. DOI: `10.1023/A:1008109111715`.

[76] M. I. Lourakis and R. Deriche, 'Camera self-calibration using the kruppa equations and the svd of the fundamental matrix: The case of varying intrinsic parameters,' INRIA, Research Report RR-3911, 2000, pp. 1–35.

[77] M. Brückner and J. Denzler, 'Active self-calibration of multi-camera systems,' in *Proceedings of 32nd Joint Pattern Recognition Symposium*, M. Goesele, S. Roth, A. Kuijper, B. Schiele and K. Schindler, Eds., Berlin, Heidelberg: Springer, 2010, pp. 31–40, ISBN: 978-3-642-15986-2. DOI: `10.1007/978-3-642-15986-2_4`.

[78] Q. Sun and D. Xu, 'Self-calibration of multi-camera networks without feature correspondence between different cameras,' *Optik*, vol. 125, no. 13, pp. 3331–3336, 2014, ISSN: 0030-4026. DOI: `10.1016/j.ijleo.2013.12.041`.

[79] F. Vasconcelos, J. P. Barreto and E. Boyer, 'Automatic camera calibration using multiple sets of pairwise correspondences,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 791–803, 2018. DOI: `10.1109/TPAMI.2017.2699648`.

[80] R. Shen, I. Cheng and A. Basu, 'Multi-camera calibration using a globe,' in *Proceedings of the 8th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras - OMNIVIS*, R. Swaminathan, V. Caglioti and A. Argyros, Eds., 2008.

[81] Z. Liu, G. Zhang, Z. Wei and J. Sun, 'A global calibration method for multiple vision sensors based on multiple targets,' *Measurement Science and Technology*, vol. 22, no. 12, 125102, 2011. DOI: `10.1088/0957-0233/22/12/125102`.

[82] R. Xia, M. Hu, J. Zhao, S. Chen and Y. Chen, 'Global calibration of multi-cameras with non-overlapping fields of view based on photogrammetry and reconfigurable target,' *Measurement Science and Technology*, vol. 29, no. 6, 065005, 2018. DOI: `10.1088/1361-6501/aab028`.

[83] F. Zhao, T. Tamaki, T. Kurita, B. Raytchev and K. Kaneda, 'Marker based simple non-overlapping camera calibration,' in *Proceedings of 2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 1180–1184. DOI: 10.1109/ICIP.2016.7532544.

[84] M. Feng, X. Jia, J. Wang, S. Feng and T. Zheng, 'Global calibration of multi-cameras based on refractive projection and ray tracing,' *Sensors*, vol. 17, no. 11, p. 2494, 2017, ISSN: 1424-8220. DOI: 10.3390/s17112494.

[85] Y. Kojima, T. Fujii and M. Tanimoto, 'New multiple-camera calibration method for a large number of cameras,' in *Videometrics VIII*, J.-A. Beraldin, S. F. El-Hakim, A. Gruen and J. S. Walton, Eds., International Society for Optics and Photonics, vol. 5665, SPIE, 2005, pp. 156–163. DOI: 10.1117/12.587700.

[86] L. Wang, F. Wu and Z. Hu, 'Multi-camera calibration with one-dimensional object under general motions,' in *Proceedings of 2007 IEEE 11th International Conference on Computer Vision (ICCV 2007)*, IEEE, 2007, pp. 1–7. DOI: 10.1109/ICCV.2007.4408994.

[87] J. A. De França, M. R. Stemmer, M. B. d. M. França and J. C. Piai, 'A new robust algorithmic for multi-camera calibration with a 1d object under general motions without prior knowledge of any camera intrinsic parameter,' *Pattern Recognition*, vol. 45, no. 10, pp. 3636–3647, 2012, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2012.04.006.

[88] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse, 'Monoslam: Real-time single camera slam,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007. DOI: 10.1109/TPAMI.2007.1049.

[89] R. A. Newcombe, S. J. Lovegrove and A. J. Davison, 'Dtam: Dense tracking and mapping in real-time,' in *Proceedings of 2011 International Conference on Computer Vision*, IEEE, 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[90] R. A. Newcombe *et al.*, 'Kinectfusion: Real-time dense surface mapping and tracking,' in *Proceedings of 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[91] A. I. Mourikis and S. I. Roumeliotis, 'A multi-state constraint kalman filter for vision-aided inertial navigation,' in *Proceedings of 2007 IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

[92] J. Engel, T. Schöps and D. Cremers, ‘Lsd-slam: Large-scale direct monocular slam,’ in *Proceedings of European Conference on Computer Vision*, D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, Eds., Springer International Publishing, 2014, pp. 834–849, ISBN: 978-3-319-10605-2. DOI: 10.1007/978-3-319-10605-2_54.

[93] J. Engel, V. Koltun and D. Cremers, ‘Direct sparse odometry,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2017. DOI: 10.1109/TPAMI.2017.2658577.

[94] F. R. Kschischang, B. J. Frey and H.-A. Loeliger, ‘Factor graphs and the sum-product algorithm,’ *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001. DOI: 10.1109/18.910572.

[95] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly and A. J. Davison, ‘Slam++: Simultaneous localisation and mapping at the level of objects,’ in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1352–1359. DOI: 10.1109/CVPR.2013.178.

[96] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker and A. Davison, ‘Elasticfusion: Dense slam without a pose graph,’ in *Proceedings of Robotics: Science and Systems XI*, L. E. Kavraki, D. Hsu and J. Buchli, Eds., RSS Foundation, Rome, Italy, 2015. DOI: 10.15607/RSS.2015.XI.001.

[97] C. G. Harris and M. Stephens, ‘A combined corner and edge detector,’ in *Proceedings of the Alvey Vision Conference, AVC*, C. J. Taylor, Ed., vol. 15, Manchester, UK: Alvey Vision Club, 1988, pp. 147–151. DOI: 10.5244/C.2.23.

[98] J. Shi and C. Tomasi, ‘Good features to track,’ in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[99] E. Rosten and T. Drummond, ‘Machine learning for high-speed corner detection,’ in *Proceedings of European Conference on Computer Vision*, A. Leonardis, H. Bischof and A. Pinz, Eds., Springer, Berlin, Heidelberg, 2006, pp. 430–443, ISBN: 978-3-540-33833-8. DOI: 10.1007/11744023_34.

[100] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos, ‘Orb-slam: A versatile and accurate monocular slam system,’ *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671.

[101] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, 'Orb: An efficient alternative to sift or surf,' in *Proceedings of 2011 International Conference on Computer Vision*, IEEE, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[102] C. Forster, M. Pizzoli and D. Scaramuzza, 'Svo: Fast semi-direct monocular visual odometry,' in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 15–22. DOI: 10.1109/ICRA.2014.6906584.

[103] V. Usenko, N. Demmel, D. Schubert, J. Stückler and D. Cremers, 'Visual-inertial mapping with non-linear factor recovery,' *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2019. DOI: 10.1109/LRA.2019.2961227.

[104] M. A. Fischler and R. C. Bolles, 'Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,' *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. DOI: 10.1145/358669.358692.

[105] M. Bloesch, S. Omari, M. Hutter and R. Siegwart, 'Robust visual inertial odometry using a direct ekf-based approach,' in *Proceedings of 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 298–304. DOI: 10.1109/IROS.2015.7353389.

[106] S. Baker and I. Matthews, 'Lucas-kanade 20 years on: A unifying framework,' *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004. DOI: 10.1023/B:VISI.0000011205.11775.fd.

[107] J. Engel, J. Stückler and D. Cremers, 'Large-scale direct slam with stereo cameras,' in *Proceedings of 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 1935–1942. DOI: 10.1109/IROS.2015.7353631.

[108] R. Wang, M. Schworer and D. Cremers, 'Stereo dso: Large-scale direct sparse visual odometry with stereo cameras,' in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3903–3911. DOI: 10.1109/ICCV.2017.421.

[109] R. Mur-Artal and J. D. Tardós, 'Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,' *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103.

[110] Z. Zhang, 'Microsoft kinect sensor and its effect,' *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012. DOI: 10.1109/MMUL.2012.24.

[111] C. Forster, Z. Zhang, M. Gassner, M. Werlberger and D. Scaramuzza, 'Svo: Semidirect visual odometry for monocular and multicamera systems,' *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016. DOI: 10.1109/TRO.2016.2623335.

[112] D. Scaramuzza, A. Martinelli and R. Siegwart, 'A flexible technique for accurate omnidirectional camera calibration and structure from motion,' in *Proceedings of Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, IEEE, 2006, pp. 45–45. DOI: 10.1109/ICVS.2006.3.

[113] D. Caruso, J. Engel and D. Cremers, 'Large-scale direct slam for omnidirectional cameras,' in *Proceedings of 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 141–148. DOI: 10.1109/IROS.2015.7353366.

[114] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler and D. Cremers, 'Omnidirectional dso: Direct sparse odometry with fisheye cameras,' *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3693–3700, 2018. DOI: 10.1109/LRA.2018.2855443.

[115] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart and P. Furgale, 'Keyframe-based visual–inertial odometry using nonlinear optimization,' *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015. DOI: 10.1177/0278364914554813.

[116] L. Von Stumberg, V. Usenko and D. Cremers, 'Direct sparse visual-inertial odometry using dynamic marginalization,' in *Proceedings of 2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2510–2517. DOI: 10.1109/ICRA.2018.8462905.

[117] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. Montiel and J. D. Tardos, 'Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam,' *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021. DOI: 10.1109/TRO.2021.3075644.

[118] A. Rosinol, M. Abate, Y. Chang and L. Carlone, 'Kimera: An open-source library for real-time metric-semantic localization and mapping,' in *Proceedings of 2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 1689–1696. DOI: 10.1109/ICRA40945.2020.9196885.

[119] A. J. Davison, 'Real-time simultaneous localisation and mapping with a single camera,' in *Proceedings Ninth IEEE International Conference on Computer Vision*, IEEE Computer Society, vol. 2, 2003, pp. 1403–1403. DOI: 10.1109/ICCV.2003.1238654.

[120] ——, ‘Scenelib 1.0.’ (2003), [Online]. Available: `https://www.doc.ic.ac.uk/~ajd/Scene/index.html` (visited on 11/03/2021).

[121] H. Kim. ‘Scenelib2.’ (2011), [Online]. Available: `https://github.com/hanmekim/SceneLib2` (visited on 11/03/2021).

[122] G. Klein and D. Murray, ‘Improving the agility of keyframe-based slam,’ in *Proceedings of European Conference on Computer Vision*, D. Forsyth, P. Torr and A. Zisserman, Eds., Springer, Berlin, Heidelberg, 2008, pp. 802–815, ISBN: 978-3-540-88688-4. DOI: `10.1007/978-3-540-88688-4_59`.

[123] ——, ‘Parallel tracking and mapping on a camera phone,’ in *Proceedings of 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, 2009, pp. 83–86. DOI: `10.1109/ISMAR.2009.5336495`.

[124] G. Klein. ‘Ptam-gpl.’ (2013), [Online]. Available: `https://github.com/Oxford-PTAM/PTAM-GPL` (visited on 11/03/2021).

[125] P. Foster. ‘Opendtam.’ (2014), [Online]. Available: `https://github.com/anuranbaka` (visited on 11/03/2021).

[126] C. Diller. ‘Kinectfusionlib.’ (2018), [Online]. Available: `https://github.com/chrdiller/KinectFusionLib` (visited on 11/03/2021).

[127] M. Li and A. I. Mourikis, ‘High-precision, consistent ekf-based visual-inertial odometry,’ *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013. DOI: `10.1177/0278364913481251`.

[128] K. Sun. ‘Msckf vio.’ (2018), [Online]. Available: `https://github.com/KumarRobotics/msckf_vio` (visited on 11/03/2021).

[129] K. Chaney. ‘Monocular msckf.’ (2018), [Online]. Available: `https://github.com/daniilidis-group/msckf_mono` (visited on 11/03/2021).

[130] J. Engel, J. Sturm and D. Cremers, ‘Semi-dense visual odometry for a monocular camera,’ in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 1449–1456. DOI: `10.1109/ICCV.2013.183`.

[131] J. Engel. ‘Lsd-slam: Large-scale direct monocular slam.’ (2014), [Online]. Available: `https://github.com/tum-vision/lsd_slam` (visited on 11/03/2021).

[132] C. Forster. ‘Svo.’ (2014), [Online]. Available: `https://github.com/uzh-rpg/rpg_svo` (visited on 11/03/2021).

[133] R. Mur-Artal. ‘Orb-slam monocular.’ (2015), [Online]. Available: `https://github.com/raulmur/ORB_SLAM` (visited on 11/03/2021).

[134] T. Whelan. ‘Elasticfusion.’ (2015), [Online]. Available: `https://github.com/mp3guy/ElasticFusion` (visited on 11/03/2021).

[135] S. Leutenegger. 'Okvis: Open keyframe-based visual-inertial slam.' (2015), [Online]. Available: `https://github.com/ethz-asl/okvis` (visited on 11/03/2021).

[136] M. Bloesch, M. Burri, S. Omari, M. Hutter and R. Siegwart, 'Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback,' *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017. DOI: `10.1177/0278364917728574`.

[137] M. Bloesch. 'Rovio.' (2015), [Online]. Available: `https://github.com/ethz-asl/rovio` (visited on 11/03/2021).

[138] R. Mur-Artal. 'Orb-slam2.' (2017), [Online]. Available: `https://github.com/raulmur/ORB_SLAM2` (visited on 11/03/2021).

[139] J. Engel. 'Dso: Direct sparse odometry.' (2017), [Online]. Available: `https://github.com/JakobEngel/dso` (visited on 11/03/2021).

[140] R. Sun. 'Vi-stereo-dso.' (2019), [Online]. Available: `https://github.com/RonaldSun/VI-Stereo-DSO` (visited on 11/03/2021).

[141] J. Zubizarreta, I. Aguinaga and J. M. M. Montiel, 'Direct sparse mapping,' *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1363–1370, 2020. DOI: `10.1109/TRO.2020.2991614`.

[142] ——, 'Dsm: Direct sparse mapping.' (2019), [Online]. Available: `https://github.com/jzubizarreta/dsm` (visited on 11/03/2021).

[143] R. Mur-Artal and J. D. Tardós, 'Visual-inertial monocular slam with map reuse,' *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017. DOI: `10.1109/LRA.2017.2653359`.

[144] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. Montiel and J. D. Tardos. 'Orb-slam3.' (2020), [Online]. Available: `https://github.com/UZ-SLAMLab/ORB_SLAM3` (visited on 11/03/2021).

[145] V. Usenko, N. Demmel, D. Schubert, J. Stückler and D. Cremers. 'Basalt.' (2019), [Online]. Available: `https://gitlab.com/VladyslavUsenko/basalt` (visited on 11/03/2021).

[146] A. Rosinol, M. Abate, Y. Chang and L. Carlone. 'Kimera.' (2019), [Online]. Available: `https://github.com/MIT-SPARK/Kimera` (visited on 11/03/2021).

[147] B. D. Lucas and T. Kanade, 'An iterative image registration technique with an application to stereo vision,' in *Proceedings of the 7th International Joint Conference on Artificial Intelligenc*, IJCAI, Vancouver, British Columbia, 1981, pp. 674–679.

[148] C. Tomasi and T. Kanade, ‘Detection and tracking of point features,’ Tech. Rep., 1991, pp. 137–154.

[149] D. G. Lowe, ‘Object recognition from local scale-invariant features,’ in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, IEEE, vol. 2, 1999, pp. 1150–1157. DOI: `10.1109/ICCV.1999.790410`.

[150] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, ‘Speeded-up robust features (surf),’ *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008, ISSN: 1077-3142. DOI: `10.1016/j.cviu.2007.09.014`.

[151] S. Leutenegger, M. Chli and R. Y. Siegwart, ‘Brisk: Binary robust invariant scalable keypoints,’ in *Proceedings of 2011 International Conference on Computer Vision*, IEEE, 2011, pp. 2548–2555. DOI: `10.1109/ICCV.2011.6126542`.

[152] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha and P. Fua, ‘Brief: Computing a local binary descriptor very fast,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2011. DOI: `10.1109/TPAMI.2011.222`.

[153] D. G. Lowe, ‘Distinctive image features from scale-invariant keypoints,’ *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. DOI: `10.1023/B:VISI.0000029664.99615.94`.

[154] B. Triggs, P. F. McLauchlan, R. I. Hartley and A. W. Fitzgibbon, ‘Bundle adjustment — a modern synthesis,’ in *International Workshop on Vision Algorithms*, B. Triggs, A. Zisserman and R. Szeliski, Eds., Springer, Berlin, Heidelberg, 1999, pp. 298–372, ISBN: 978-3-540-44480-0. DOI: `10.1007/3-540-44480-7_21`.

[155] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman and W. T. Freeman, ‘Discovering objects and their location in images,’ in *Proceedings of Tenth IEEE International Conference on Computer Vision (ICCV’05)*, IEEE, vol. 1, 2005, pp. 370–377. DOI: `10.1109/ICCV.2005.77`.

[156] D. Devarajan and R. J. Radke, ‘Calibrating distributed camera networks using belief propagation,’ *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 221–221, 2007. DOI: `10.1155/2007/60696`.

[157] D. Devarajan, R. J. Radke and H. Chung, ‘Distributed metric calibration of ad hoc camera networks,’ *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 3, pp. 380–403, 2006. DOI: `10.1145/1167935.1167939`.

[158] D. Devarajan and R. J. Radke, ‘Distributed metric calibration of large camera networks,’ in *Proceedings of the First Workshop on Broadband Advanced Sensor Networks*, vol. 3, 2004, pp. 5–24.

[159] D. Makris, T. Ellis and J. Black, ‘Bridging the gaps between cameras,’ in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR 2004)*, IEEE, vol. 2, 2004. DOI: 10.1109/CVPR.2004.1315165.

[160] D. Marinakis and G. Dudek, ‘Topology inference for a vision-based sensor network,’ in *Proceedings of the 2nd Canadian Conference on Computer and Robot Vision (CRV’05)*, IEEE, 2005, pp. 121–128. DOI: 10.1109/CRV.2005.81.

[161] X. Zou, B. Bhanu, B. Song and A. K. Roy-Chowdhury, ‘Determining topology in a distributed camera network,’ in *Proceedings of the 2007 IEEE International Conference on Image Processing*, IEEE, vol. 5, 2007, pp. 133–136. DOI: 10.1109/ICIP.2007.4379783.

[162] R. Farrell, D. Doermann and L. S. Davis, ‘Learning higher-order transition models in medium-scale camera networks,’ in *Proceedings of the 2007 IEEE 11th International Conference on Computer Vision*, IEEE, 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4409203.

[163] R. Farrell and L. S. Davis, ‘Decentralized discovery of camera network topology,’ in *Proceedings of the 2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, IEEE, 2008, pp. 1–10. DOI: 10.1109/ICDSC.2008.4635696.

[164] P. Kulkarni, P. Shenoy and D. Ganesan, ‘Approximate initialization of camera sensor networks,’ in *Proceedings of the European Conference on Wireless Sensor Networks*, K. Langendoen and T. Voigt, Eds., Springer, 2007, pp. 67–82, ISBN: 978-3-540-69830-2. DOI: 10.1007/978-3-540-69830-2_5.

[165] S. Avidan, Y. Moses and Y. Moses, ‘Centralized and distributed multi-view correspondence,’ *International Journal of Computer Vision*, vol. 71, no. 1, pp. 49–69, 2007. DOI: 10.1007/s11263-005-4888-y.

[166] Z. Cheng, D. Devarajan and R. J. Radke, ‘Determining vision graphs for distributed camera networks using feature digests,’ *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 220–220, 2007. DOI: 10.1155/2007/57034.

[167] A. Van Den Hengel, A. Dick, H. Detmold, A. Cichowski and R. Hill, ‘Finding camera overlap in large surveillance networks,’ in *Proceedings of the 2007 Asian Conference on Computer Vision (ACCV 2007)*, Y. Yagi, S. B. Kang, I. S. Kweon and H. Zha, Eds., Springer, Berlin, Heidelberg, 2007, pp. 375–384, ISBN: 978-3-540-76386-4. DOI: 10.1007/978-3-540-76386-4_35.

[168] Z. Mandel, I. Shimshoni and D. Keren, 'Multi-camera topology recovery from coherent motion,' in *Proceedings of the 2007 First ACM/IEEE International Conference on Distributed Smart Cameras*, IEEE, 2007, pp. 243–250. DOI: `10.1109/ICDSC.2007.4357530`.

[169] E. B. Ermis, P. Clarot, P.-M. Jodoin and V. Saligrama, 'Activity based matching in distributed camera networks,' *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2595–2613, 2010. DOI: `10.1109/TIP.2010.2052824`.

[170] L. Esterle, P. R. Lewis, X. Yao and B. Rinner, 'Socio-economic vision graph generation and handover in distributed smart camera networks,' *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 2, p. 20, 2014. DOI: `10.1145/2530001`.

[171] R. O. Saber and R. M. Murray, 'Consensus protocols for networks of dynamic agents,' in *Proceedings of the 2003 American Control Conference*, vol. 2, 2003, pp. 951–956. DOI: `10.1109/ACC.2003.1239709`.

[172] W. Ren and R. W. Beard, 'Consensus seeking in multiagent systems under dynamically changing interaction topologies,' *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005. DOI: `10.1109/TAC.2005.846556`.

[173] M. G. Rabbat, R. D. Nowak and J. A. Bucklew, 'Generalized consensus computation in networked systems with erasure links,' in *IEEE 6th Workshop on Signal Processing Advances in Wireless Communications*, IEEE, 2005, pp. 1088–1092. DOI: `10.1109/SPAWC.2005.1506308`.

[174] F. Xiao and L. Wang, 'Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays,' *IEEE Transactions on Automatic Control*, vol. 53, no. 8, pp. 1804–1816, 2008. DOI: `10.1109/TAC.2008.929381`.

[175] Y. G. Sun, L. Wang and G. Xie, 'Average consensus in networks of dynamic agents with switching topologies and multiple time-varying delays,' *Systems & Control Letters*, vol. 57, no. 2, pp. 175–183, 2008, ISSN: 0167-6911. DOI: `10.1016/j.sysconle.2007.08.009`.

[176] P. Lin and Y. Jia, 'Average consensus in networks of multi-agents with both switching topology and coupling time-delay,' *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 1, pp. 303–313, 2008, ISSN: 0378-4371. DOI: `10.1016/j.physa.2007.08.040`.

[177] Y. Chen, R. Tron, A. Terzis and R. Vidal, 'Corrective consensus: Converging to the exact average,' in *Proceedings of the 2010 49th IEEE Conference on Decision and Control (CDC)*, IEEE, 2010, pp. 1221–1228. DOI: `10.1109/CDC.2010.5717925`.

[178] Y. Su and J. Huang, 'Two consensus problems for discrete-time multi-agent systems with switching network topology,' *Automatica*, vol. 48, no. 9, pp. 1988–1997, 2012, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2012.03.029.

[179] F. Chen, Y. Cao, W. Ren *et al.*, 'Distributed average tracking of multiple time-varying reference signals with bounded derivatives,' *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3169–3174, 2012. DOI: 10.1109/TAC.2012.2199176.

[180] X. Xu, S. Chen, W. Huang and L. Gao, 'Leader-following consensus of discrete-time multi-agent systems with observer-based protocols,' *Neurocomputing*, vol. 118, pp. 334–341, 2013, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2013.02.023.

[181] G. Wen, Z. Duan, G. Chen and W. Yu, 'Consensus tracking of multi-agent systems with lipschitz-type node dynamics and switching topologies,' *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 499–511, 2014. DOI: 10.1109/TCSI.2013.2268091.

[182] G. S. Seyboth, D. V. Dimarogonas and K. H. Johansson, 'Event-based broadcasting for multi-agent average consensus,' *Automatica*, vol. 49, no. 1, pp. 245–252, 2013, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2012.08.042.

[183] C. Nowzari and J. Cortés, 'Distributed event-triggered coordination for average consensus on weight-balanced digraphs,' *Automatica*, vol. 68, pp. 237–244, 2016, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2016.01.069.

[184] H. Li, G. Chen, T. Huang, Z. Dong, W. Zhu and L. Gao, 'Event-triggered distributed average consensus over directed digital networks with limited communication bandwidth,' *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 3098–3110, 2016. DOI: 10.1109/TCYB.2015.2496977.

[185] Z. Zuo and L. Tie, 'Distributed robust finite-time nonlinear consensus protocols for multi-agent systems,' *International Journal of Systems Science*, vol. 47, no. 6, pp. 1366–1375, 2016. DOI: 10.1080/00207721.2014.925608.

[186] C. N. Hadjicostis, N. H. Vaidya and A. D. Dominguez-Garcia, 'Robust distributed average consensus via exchange of running sums,' *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2016. DOI: 10.1109/TAC.2015.2471695.

[187] A. Jadbabaie, J. Lin and A. S. Morse, 'Coordination of groups of mobile autonomous agents using nearest neighbor rules,' *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003. DOI: 10.1109/TAC.2003.812781.

[188] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen and O. Shochet, 'Novel type of phase transition in a system of self-driven particles,' *Physical Review Letters*, vol. 75, no. 6, pp. 1226–1229, 1995. DOI: `10.1103/PhysRevLett.75.1226`.

[189] A. V. Savkin, 'Coordinated collective motion of groups of autonomous mobile robots: Analysis of vicsek's model,' *IEEE Transactions on Automatic Control*, vol. 49, no. 6, pp. 981–982, 2004. DOI: `10.1109/TAC.2004.829621`.

[190] Z. Lin, M. Broucke and B. Francis, 'Local control strategies for groups of mobile autonomous agents,' *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 622–629, 2004. DOI: `10.1109/TAC.2004.825639`.

[191] A. Muhammad and M. Egerstedt, 'Connectivity graphs as models of local interactions,' *Applied Mathematics and Computation*, vol. 168, no. 1, pp. 243–269, 2005, ISSN: 0096-3003. DOI: `10.1016/j.amc.2004.08.039`.

[192] J. Cortes, S. Martinez, T. Karatas and F. Bullo, 'Coverage control for mobile sensing networks,' *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004. DOI: `10.1109/TRA.2004.824698`.

[193] R. Olfati-Saber and J. S. Shamma, 'Consensus filters for sensor networks and distributed sensor fusion,' in *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 6698–6703. DOI: `10.1109/CDC.2005.1583238`.

[194] D. P. Spanos, R. Olfati-Saber and R. M. Murray, 'Approximate distributed kalman filtering in sensor networks with quantifiable performance,' in *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks*, IEEE, 2005, pp. 133–139. DOI: `10.1109/IPSN.2005.1440912`.

[195] R. Olfati-Saber, 'Distributed kalman filter with embedded consensus filters,' in *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 8179–8184. DOI: `10.1109/CDC.2005.1583486`.

[196] ——, 'Distributed kalman filtering for sensor networks,' in *Proceedings of the 2007 46th IEEE Conference on Decision and Control*, IEEE, 2007, pp. 5492–5498. DOI: `10.1109/CDC.2007.4434303`.

[197] J. Pearl, 'Fusion, propagation, and structuring in belief networks,' *Artificial intelligence*, vol. 29, no. 3, pp. 241–288, 1986, ISSN: 0004-3702. DOI: `10.1016/0004-3702(86)90072-X`.

[198] K. P. Murphy, Y. Weiss and M. I. Jordan, 'Loopy belief propagation for approximate inference: An empirical study,' in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., Stockholm, Sweden, 1999, pp. 467–475, ISBN: 1-55860-614-9. DOI: `10.5555/2073796.2073849`.

[199] J. S. Yedidia, W. T. Freeman, Y. Weiss *et al.*, 'Generalized belief propagation,' in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, T. Leen, T. Dietterich and V. Tresp, Eds., vol. 13, MIT Press, 2000, pp. 689–695. DOI: `10.5555/3008751.3008848`.

[200] Y. Weiss and W. T. Freeman, 'Correctness of belief propagation in gaussian graphical models of arbitrary topology,' vol. 12, S. Solla, T. Leen and K. Müller, Eds., pp. 673–679, 1999. DOI: `10.5555/3009657.3009753`.

[201] C. C. Moallemi and B. Van Roy, 'Consensus propagation,' *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4753–4766, 2006. DOI: `10.1109/TIT.2006.883539`.

[202] R. Olfati-Saber, E. Franco, E. Frazzoli and J. Shamma, 'Belief consensus and distributed hypothesis testing in sensor networks,' *Networked Embedded Sensing and Control*, pp. 169–182, 2006. DOI: `10.1007/11533382_11`.

[203] A. T. Ihler, J. W. Fisher, R. L. Moses and A. S. Willsky, 'Nonparametric belief propagation for self-localization of sensor networks,' *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, 2005. DOI: `10.1109/JSAC.2005.843548`.

[204] J. K. Johnson, D. M. Malioutov and A. S. Willsky, 'Walk-sum interpretation and analysis of gaussian belief propagation,' *Proceedings of the 18th International Conference on Neural Information Processing Systems*, vol. 18, pp. 579–586, 2006. DOI: `10.5555/2976248.2976321`.

[205] D. M. Malioutov, J. K. Johnson and A. S. Willsky, 'Walk-sums and belief propagation in gaussian graphical models,' *Journal of Machine Learning Research*, vol. 7, pp. 2031–2064, 2006.

[206] Q. Su and Y.-C. Wu, 'Convergence analysis of the variance in gaussian belief propagation,' *IEEE Transactions on Signal Processing*, vol. 62, no. 19, pp. 5119–5131, 2014. DOI: `10.1109/TSP.2014.2345635`.

[207] ——, 'On convergence conditions of gaussian belief propagation,' *IEEE Transactions on Signal Processing*, vol. 63, no. 5, pp. 1144–1155, 2015. DOI: `10.1109/TSP.2015.2389755`.

[208] J. Du, S. Ma, Y.-C. Wu, S. Kar and J. M. Moura, 'Convergence analysis of distributed inference with vector-valued gaussian belief propagation,' *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6302–6339, 2017.

[209] ——, 'Convergence analysis of the information matrix in gaussian belief propagation,' in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4074–4078. DOI: 10.1109/ICASSP.2017.7952922.

[210] E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman and A. S. Willsky, 'Nonparametric belief propagation,' *Communications of the ACM*, vol. 53, no. 10, pp. 95–103, 2010. DOI: 10.1145/1831407.1831431.

[211] V. Savic, H. Wymeersch, F. Penna and S. Zazo, 'Optimized edge appearance probability for cooperative localization based on tree-reweighted nonparametric belief propagation,' in *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, pp. 3028–3031. DOI: 10.1109/ICASSP.2011.5946296.

[212] O. Shental, P. H. Siegel, J. K. Wolf, D. Bickson and D. Dolev, 'Gaussian belief propagation solver for systems of linear equations,' in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, IEEE, 2008, pp. 1863–1867.

[213] D. Bickson, E. Yom-Tov and D. Dolev, 'A gaussian belief propagation solver for large scale support vector machines,' *arXiv preprint arXiv:0810.1648*, 2008. DOI: 10.48550/arXiv.0810.1648.

[214] D. Bickson, O. Shental and D. Dolev, 'Distributed kalman filter via gaussian belief propagation,' in *Proceedings of the 2008 46th Annual Allerton Conference on Communication, Control, and Computing*, IEEE, 2008, pp. 628–635. DOI: 10.1109/ALLERTON.2008.4797617.

[215] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, 'Distributed optimization and statistical learning via the alternating direction method of multipliers,' *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011. DOI: 10.1561/2200000016.

[216] N. Parikh and S. Boyd, 'Proximal algorithms,' *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014. DOI: 10.1561/2400000003.

[217] P. L. Combettes and J.-C. Pesquet, 'Proximal splitting methods in signal processing,' in *Fixed-point algorithms for inverse problems in science and engineering*, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke and H. Wolkowicz, Eds., New York, NY: Springer, 2011, pp. 185–212, ISBN: 978-1-4419-9569-8. DOI: 10.1007/978-1-4419-9569-8_10.

[218] S. Setzer, 'Split bregman algorithm, douglas-rachford splitting and frame shrinkage,' in *Proceedings of the International Conference on Scale Space and Variational Methods in Computer Vision*, X.-C. Tai, K. Mørken, M. Lysaker and K.-A. Lie, Eds., Springer, Berlin, Heidelberg, 2009, pp. 464–476, ISBN: 978-3-642-02256-2. DOI: `10.1007/978-3-642-02256-2_39`.

[219] I. D. Schizas, A. Ribeiro and G. B. Giannakis, 'Consensus in ad hoc wsns with noisy links—part i: Distributed estimation of deterministic signals,' *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008. DOI: `10.1109/TSP.2007.906734`.

[220] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis and A. Ribeiro, 'Consensus in ad hoc wsns with noisy links—part ii: Distributed estimation and smoothing of random signals,' *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1650–1666, 2008. DOI: `10.1109/TSP.2007.908943`.

[221] A. Eriksson and M. Isaksson, 'Pseudoconvex proximal splitting for l-infinity problems in multiview geometry,' in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 4066–4073. DOI: `10.1109/CVPR.2014.518`.

[222] A. Eriksson, M. Isaksson and T.-J. Chin, 'High breakdown bundle adjustment,' in *Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2015, pp. 310–317. DOI: `10.1109/WACV.2015.48`.

[223] A. Eriksson, J. Bastian, T.-J. Chin and M. Isaksson, 'A consensus-based framework for distributed bundle adjustment,' in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1754–1762. DOI: `10.1109/WACV.2015.48`.

[224] E. Elhamifar and R. Vidal, 'Distributed calibration of camera sensor networks,' in *Proceedings of the 2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, IEEE, 2009, pp. 1–7. DOI: `10.1109/ICDSC.2009.5289397`.

[225] W. E. Mantzel, H. Choi and R. G. Baraniuk, 'Distributed camera network localization,' in *Proceedings of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers.*, IEEE, vol. 2, 2004, pp. 1381–1386. DOI: `10.1109/ACSSC.2004.1399380`.

[226] A. Barton-Sweeney, D. Lymberopoulos and A. Savvides, 'Sensor localization and camera calibration in distributed camera sensor networks,' in *Proceedings of the 2006 3rd International Conference on Broadband Communications, Networks and Systems*, IEEE, 2006, pp. 1–10. DOI: `10.1109/BROADNETS.2006.4374301`.

[227] G. Kurillo, Z. Li and R. Bajcsy, 'Wide-area external multi-camera calibration using vision graphs and virtual calibration object,' in *Proceedings of the 2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, IEEE, 2008, pp. 1–9. DOI: 10.1109/ICDSC.2008.4635695.

[228] H. Medeiros, H. Iwaki and J. Park, 'Online distributed calibration of a large network of wireless cameras using dynamic clustering,' in *Proceedings of the 2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, IEEE, 2008, pp. 1–10. DOI: 10.1109/ICDSC.2008.4635698.

[229] T. Pollok and E. Monari, 'A visual slam-based approach for calibration of distributed camera networks,' in *Proceedings of the 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, 2016, pp. 429–437. DOI: 10.1109/AVSS.2016.7738081.

[230] L. Riazuelo, J. Civera and J. M. Montiel, 'C2tam: A cloud framework for cooperative tracking and mapping,' *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014, ISSN: 0921-8890. DOI: 10.1016/j.robot.2013.11.007.

[231] J. Dong, E. Nelson, V. Indelman, N. Michael and F. Dellaert, 'Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach,' in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 5807–5814. DOI: 10.1109/ICRA.2015.7140012.

[232] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese and C. Hertzberg, 'Hierarchical optimization on manifolds for online 2d and 3d mapping,' in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 273–278. DOI: 10.1109/ROBOT.2010.5509407.

[233] K. Ni and F. Dellaert, 'Multi-level submap based slam using nested dissection,' in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 2558–2565. DOI: 10.1109/IROS.2010.5650197.

[234] L. Zhao, S. Huang and G. Dissanayake, 'Linear slam: A linear solution to the feature-based and pose graph slam based on submap joining,' in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 24–30. DOI: 10.1109/IROS.2013.6696327.

[235] A. Birk and S. Carpin, 'Merging occupancy grid maps from multiple robots,' *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384–1397, 2006. DOI: 10.1109/JPROC.2006.876965.

[236] M. T. Lazaro, L. M. Paz, P. Pinies, J. A. Castellanos and G. Grisetti, 'Multi-robot slam using condensed measurements,' in *Proceeedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 1069–1076. DOI: 10.1109/IROS.2013.6696483.

[237] R. Aragues, J. Cortes and C. Sagues, 'Distributed consensus on robot networks for dynamically merging feature-based maps,' *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 840–854, 2012. DOI: 10.1109/TRO.2012.2192012.

[238] E. D. Nerurkar, S. I. Roumeliotis and A. Martinelli, 'Distributed maximum a posteriori estimation for multi-robot cooperative localization,' in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 1402–1409. DOI: 10.1109/ROBOT.2009.5152398.

[239] L. Paull, G. Huang, M. Seto and J. J. Leonard, 'Communication-constrained multi-auv cooperative slam,' in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 509–516. DOI: 10.1109/ICRA.2015.7139227.

[240] T. Cieslewski, S. Choudhary and D. Scaramuzza, 'Data-efficient decentralized visual slam,' in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2466–2473. DOI: 10.1109/ICRA.2018.8461155.

[241] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone and G. Beltrame, 'Door-slam: Distributed, online, and outlier resilient slam for robotic teams,' *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1656–1663, 2020. DOI: 10.1109/LRA.2020.2967681.

[242] D. Bickson, 'Gaussian belief propagation: Theory and application,' Ph.D. dissertation, The Hebrew University of Jerusalem, 2008. DOI: 10.48550/arXiv.0811.2518.

[243] L. Wang, W. Wang, C. Shen and F. Duan, 'A convex relaxation optimization algorithm for multi-camera calibration with 1d objects,' *Neurocomputing*, vol. 215, pp. 82–89, 2016, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2015.07.158.

[244] B. Triggs, 'Autocalibration from planar scenes,' in *Proceedings of the European Conference on Computer Vision*, H. Burkhardt and B. Neumann, Eds., Springer, Berlin, Heidelberg, 1998, pp. 89–105, ISBN: 978-3-540-69354-3. DOI: 10.1007/BFb0055661.

[245] S. Ramalingam and P. Sturm, 'A unifying model for camera calibration,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1309–1319, 2017. DOI: 10.1109/TPAMI.2016.2592904.

[246] Y. Bastanlar, 'A simplified two-view geometry based external calibration method for omnidirectional and ptz camera pairs,' *Pattern Recognition Letters*, vol. 71, pp. 1–7, 2016, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2015.11.013.

[247] A. Zaharescu, R. Horaud, R. Ronfard and L. Lefort, 'Multiple camera calibration using robust perspective factorization,' in *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, IEEE, 2006, pp. 504–511. DOI: 10.1109/3DPVT.2006.100.

[248] M. Zhu and S. Martinez, 'Discrete-time dynamic average consensus,' *Automatica*, vol. 46, no. 2, pp. 322–329, 2010, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2009.10.021.

[249] J. Knuth and P. Barooah, 'Distributed collaborative 3d pose estimation of robots from heterogeneous relative measurements: An optimization on manifold approach,' *Robotica*, vol. 33, no. 7, pp. 1507–1535, 2015. DOI: 10.1017/S0263574714000794.

[250] J. Thunberg, J. Goncalves and X. Hu, 'Consensus and formation control on se (3) for switching topologies,' *Automatica*, vol. 66, pp. 109–121, 2016, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2015.12.035.

[251] L. Luft, T. Schubert, S. I. Roumeliotis and W. Burgard, 'Recursive decentralized localization for multi-robot systems with asynchronous pairwise communication,' *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1152–1167, 2018. DOI: 10.1177/0278364918760698.

[252] H. Cui, S. Shen and Z. Hu, 'Tracks selection for robust, efficient and scalable large-scale structure from motion,' *Pattern Recognition*, vol. 72, pp. 341–354, 2017, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2017.08.002.

[253] R. Munoz-Salinas, M. J. Marin-Jimenez and R. Medina-Carnicer, 'Spm-slam: Simultaneous localization and mapping with squared planar markers,' *Pattern Recognition*, vol. 86, pp. 156–171, 2019, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2018.09.003.

[254] R. Munoz-Salinas, M. J. Marin-Jimenez, E. Yeguas-Bolivar and R. Medina-Carnicer, 'Mapping and localization from planar markers,' *Pattern Recognition*, vol. 73, pp. 158–171, 2018.

[255] D. W. Marquardt, 'An algorithm for least-squares estimation of nonlinear parameters,' *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. DOI: 10.1137/0111030.

[256] T. D. Barfoot and P. T. Furgale, ‘Associating uncertainty with three-dimensional poses for use in estimation problems,’ *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014. DOI: 10.1109/TRO.2014.2298059.

[257] P. H. Torr and A. Zisserman, ‘Feature based methods for structure and motion estimation,’ in *Proceedings of the 1999 International Workshop on Vision Algorithms*, B. Triggs, A. Zisserman and R. Szeliski, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 1999, pp. 278–294, ISBN: 978-3-540-44480-0. DOI: 10.1007/3-540-44480-7_19.

[258] O. Demetz, ‘Feature invariance versus change estimation in variational motion estimation,’ Ph.D. dissertation, University of Saarland, 2015. DOI: 10.22028/D291-26623.

[259] C. Liu, J. Yuen and A. Torralba, ‘Sift flow: Dense correspondence across scenes and its applications,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 978–994, 2010. DOI: 10.1109/TPAMI.2010.147.

[260] R. Zabih and J. Woodfill, ‘Non-parametric local transforms for computing visual correspondence,’ in *Proceedings of European Conference on Computer Vision*, J.-O. Eklundh, Ed., Springer, Berlin, Heidelberg, 1994, pp. 151–158, ISBN: 978-3-540-48400-4. DOI: 10.1007/BFb0028345.

[261] G. Silveira and E. Malis, ‘Real-time visual tracking under arbitrary illumination changes,’ in *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–6. DOI: 10.1109/CVPR.2007.382993.

[262] H. Bristow and S. Lucey, ‘In defense of gradient-based alignment on densely sampled sparse features,’ in *Dense Image Correspondences for Computer Vision*, T. Hassner and C. Liu, Eds., Springer International Publishing, 2016, pp. 135–152, ISBN: 978-3-319-23048-1. DOI: 10.1007/978-3-319-23048-1_7.

[263] L. Sevilla-Lara and E. Learned-Miller, ‘Distribution fields for tracking,’ in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 1910–1917. DOI: 10.1109/CVPR.2012.6247891.

[264] L. Sevilla-Lara, D. Sun, E. G. Learned-Miller and M. J. Black, ‘Optical flow estimation with channel constancy,’ in *Proceedings of the European Conference on Computer Vision*, D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, Eds., Springer International Publishing, 2014, pp. 423–438, ISBN: 978-3-319-10590-1. DOI: 10.1007/978-3-319-10590-1_28.

[265] H. Alismail, M. Kaess, B. Browning and S. Lucey, 'Direct visual odometry in low light using binary descriptors,' *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 444–451, 2016. DOI: `10.1109/LRA.2016.2635686`.

[266] S. Park, T. Schöps and M. Pollefeys, 'Illumination change robustness in direct visual slam,' in *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 4523–4530. DOI: `10.1109/ICRA.2017.7989525`.

[267] M. Peris, S. Martull, A. Maki, Y. Ohkawa and K. Fukui, 'Towards a simulation driven stereo vision system,' in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, IEEE, 2012, pp. 1038–1042.

[268] A. Geiger, P. Lenz and R. Urtasun, 'Are we ready for autonomous driving? the kitti vision benchmark suite,' in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 3354–3361. DOI: `10.1109/CVPR.2012.6248074`.

[269] D. Bickson, D. Dolev, O. Shental, P. H. Siegel and J. K. Wolf, 'Gaussian belief propagation based multiuser detection,' in *Proceedings of the 2008 IEEE International Symposium on Information Theory*, IEEE, 2008, pp. 1878–1882. DOI: `10.1109/ISIT.2008.4595314`.

[270] O. Shental, P. H. Siegel, J. K. Wolf, D. Bickson and D. Dolev, 'Gaussian belief propagation solver for systems of linear equations,' in *Proceedings of the 2008 IEEE International Symposium on Information Theory*, IEEE, 2008, pp. 1863–1867. DOI: `10.1109/ISIT.2008.4595311`.

[271] V. Savic and S. Zazo, 'Cooperative localization in mobile networks using non-parametric variants of belief propagation,' *Ad Hoc Networks*, vol. 11, no. 1, pp. 138–150, 2013, ISSN: 1570-8705. DOI: `10.1016/j.adhoc.2012.04.012`.

[272] V. Savic and H. Wymeersch, 'Simultaneous localization and tracking via real-time nonparametric belief propagation,' in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 5180–5184. DOI: `10.1109/ICASSP.2013.6638650`.

[273] A. F. Garcia-Fernandez, L. Svensson and S. Särkkä, 'Cooperative localization using posterior linearization belief propagation,' *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 832–836, 2017. DOI: `10.1109/TVT.2017.2734683`.

[274] Z. Hosseinidoust, D. Giannacopoulos and W. J. Gross, 'Gpu optimization and implementation of gaussian belief propagation algorithm,' in *Proceedings of the 2016 IEEE Conference on Electromagnetic Field Computation (CEFC)*, IEEE, 2016, pp. 1–5. DOI: `10.1109/CEFC.2016.7816128`.

[275] Y. El-Kurdi, D. Fernández, W. J. Gross and D. D. Giannacopoulos, 'Acceleration of the finite-element gaussian belief propagation solver using minimum residual techniques,' *IEEE Transactions on Magnetics*, vol. 52, no. 3, pp. 1–4, 2015. DOI: `10.1109/TMAG.2015.2487683`.

[276] M. J. Wainwright, T. S. Jaakkola and A. S. Willsky, 'Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching,' in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, C. M. Bishop and B. J. Frey, Eds., ser. Proceedings of Machine Learning Research, PMLR, vol. R4, 2003, pp. 308–315.

[277] H. Wymeersch, F. Penna and V. Savić, 'Uniformly reweighted belief propagation: A factor graph approach,' in *Proceedings of the 2011 IEEE International Symposium on Information Theory*, IEEE, 2011, pp. 2000–2004. DOI: `10.1109/ISIT.2011.6033905`.

[278] D. J. Watts and S. H. Strogatz, 'Collective dynamics of 'small-world' networks,' *Nature*, vol. 393, no. 6684, pp. 440–442, 1998. DOI: `10.1038/30918`.

[279] F. Meyer *et al.*, 'Message passing algorithms for scalable multitarget tracking,' *Proceedings of the IEEE*, vol. 106, no. 2, pp. 221–259, 2018. DOI: `10.1109/JPROC.2018.2789427`.

[280] E. Leitinger, F. Meyer, F. Hlawatsch, K. Witrisal, F. Tufvesson and M. Z. Win, 'A belief propagation algorithm for multipath-based slam,' *IEEE Transactions on Wireless Communications*, vol. 18, no. 12, pp. 5613–5629, 2019. DOI: `10.1109/TWC.2019.2937781`.