



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Head Office: Università degli Studi di Padova
Department of Information Engineering (DEI)

Ph.D. Course in Information Engineering
Curriculum in Information and Communication Technologies
XXXVII Cycle

Robust Pose Graph Optimization for Calibration and Autonomous Robots Navigation

Coordinator: Prof. Fabio Vandin
Supervisor: Prof. Alberto Pretto
Ph.D. Student: Emilio Olivastri

ABSTRACT

A Pose Graph (PG) is a powerful mathematical framework used to formulate a wide range of robotic problems. These problems can be expressed as Maximum a Posteriori (MAP) estimation tasks, which are efficiently solved using Non-Linear Least Squares (NLLS) optimization. The Pose Graph’s versatility extends across various state estimation problems, including Simultaneous Localization and Mapping (SLAM), Structure-from-Motion (SfM), optimal control, tracking, extrinsic sensor calibration, and multimodal sensor fusion. This broad applicability positions Pose Graph Optimization (PGO) as a fundamental tool for spatial perception—a critical skill that enables robots to ground sensor data into structured 3D representations, allowing them to perceive, model, and understand their surroundings effectively.

Among the diverse applications of PGO, extrinsic calibration and SLAM are particularly crucial and interrelated. Extrinsic calibration involves estimating relative poses between sensors or various robot components, such as determining the pose between a robot base and an arm’s end effector mounted on it. Accurate extrinsic calibration is critical for sensor fusion and reliable autonomous task execution. Instead, SLAM is the problem of incrementally building a map of an unknown environment while simultaneously estimating the robot’s own pose using the perceived sensor information. This capability is essential for real-world deployment, allowing autonomous navigation and interaction within unfamiliar surroundings.

This thesis emphasizes the versatility of PGs through two significant contributions. Firstly, it demonstrates their application in scarcely explored domains, specifically Hand-Eye (HE) calibration, introducing a general framework for estimating spatial relationships between a common reference frame and multiple cameras within robotic workspaces.

Secondly, this research addresses a critical limitation of PGO: its inherent assumption that the data used in the optimization does not contain spurious measurements, which often does not hold true. Perception errors, known as outliers, frequently occur in perceptually aliased environments such as underwater, cave systems, and urban areas, where visually similar locations lead to incorrect correspondences. Because PGO is highly sensitive to incorrect constraints, even a small fraction of erroneous correspondences can severely distort the estimated trajectory and compromise system reliability. Consequently, this thesis tackles this limitation specifically within the SLAM context. It introduces novel Robust Pose Graph Optimization (RPGO) frameworks that mitigate the adverse effects of outliers through adaptive weighting schemes, consistency checks, and multi-horizon reasoning. By enhancing the resilience of PGO against erroneous constraints, the presented approaches contribute significantly to achieving reliable long-term autonomy in challenging exploration scenarios.

ACKNOWLEDGEMENTS

First, I want to thank my supervisor, Alberto Pretto, for encouraging me to pursue a Ph.D. and guiding me throughout the process. I know I haven't been the easiest student, as we both have strong opinions, but these discussions helped me grow as a person and researcher. You always allowed me the freedom to pursue my own interests, and for this, I truly want to thank you. I want to thank Prof. Emanuele Menegatti. As the Head of the Lab, he was always responsive to our requests, encouraged us to push further, and always open to joke around. As a professor, he was also my first contact to the world of robotics, a field I have come to love.

Of course, I also want to thank my brothers in arms during my Ph.D. The research path is not an easy one, and striving to reach those objectives alone would have been even harder. I am deeply grateful to my fellow Ph.D. students, Post-Docs and Professors who accompanied me over these three years. A special thanks to Daniele Evangelista, Matteo Terreran, and Stefano Ghidoni, who were not only irreplaceable friends but also invaluable guides. Daniele taught me to take things more lightly and handle negative situations with grace, Matteo inspired me with his passion and dedication to research, all while taking responsibility for the team, while Stefano was always willing to embark in new adventures with me. I also want to extend a heartfelt thanks to Leonardo Barcellona, Alberto Bacchin, and Alberto Gottardi—going through these three years together, supporting each other during the hard moments, made everything better. I'll never forget it.

I also need to thank Niko Sünderhauf for welcoming me as a visiting researcher at Queensland University of Technology. Thanks to you, I had the opportunity to meet incredible people many great friends inside and outside the lab. I am especially grateful to Dr. Krishan Rana, who was a true inspiration during my time at QCR. His incredible passion and unique perspective opened a new world for me. I also want to include Alejandro Fontán Villacampa in my thanks. Our brief discussions about SLAM often turned into hour-long debates, and I cherished every moment of them. Thanks to him, I discovered a new passion for diving. I must also thank my friend Nicolas Chapman for being such an amazing friend and researcher. From work to personal life, I could talk to you about anything, and your presence made everything far more enjoyable.

Last but certainly not least, I want to thank Scarlett Georgina Raine, the most important person I met during this time. She is a brilliant researcher and an extraordinary individual whom I deeply admire for pursuing her dreams despite all adversities. Meeting her gave me the motivation I needed to strive harder for my own dreams and happiness.

Finally, I need to thank my family because, without them, I wouldn't have been able to achieve as much as I did. To my mum, Francis Ginart, who always pushed me to do more and supported me in every possible way. To my dad, Emilio Olivastri, who taught me to never give up and to become the person I am proud to be today. To my sister Maria Laura, who was and is always there for me—no matter what, I know I can always count on her. To my brother-in-law, James Robert Robinson, who has become the brother I never had. From teasing each other to having deep conversations about important matters, your presence has meant so much to me. To my nephews,

Leonard and Matteo, for always making me proud and loving me despite my constant pranks and not letting you win at any games. I also want to thank my uncles, Antonio Ginart and Rosana Esteller, for being role models and, even from a distance, always keeping an eye on me and supporting me throughout my life. Lastly, I need to thank my cousin, Tony Ginart, for unknowingly being my rival. One of the reasons I pushed so hard was that I didn't want to lose to him, but even more so because I wanted to make him proud.

Table of Contents

Abstract	i
Acknowledgements	ii
LIST OF FIGURES	vii
LIST OF TABLES	ix
GLOSSARY	xi
1 INTRODUCTION	1
1.1 Main Contributions	4
1.2 List of Publications	6
1.3 Open-Source Contributions	7
2 PRELIMINARIES	9
2.1 Dynamic Bayesian Network Model	11
2.2 Pose Graph Model	13
2.3 Non-Linear Least Squares	16
2.3.1 Uncertainty and Non-Linear Least Squares	21
2.3.2 Leveraging Sparsity	24
2.3.3 Re-weighted Non-Linear Least Squares	26
2.3.4 Optimization on Manifolds	27
2.4 Factor Graphs for Robotics	30
3 A GRAPH-BASED OPTIMIZATION FRAMEWORK FOR HAND-EYE CALIBRATION FOR MULTI-CAMERA SETUPS	35
3.1 Motivation	35
3.2 Related Work	37
3.3 Method	38
3.3.1 Eye-on-Base Calibration for Single Camera	38
3.3.2 Eye-on-Base Calibration for Multi-Camera	41
3.4 Evaluation	44
3.4.1 Experiments with Synthetic Data	45
3.4.2 Experiments on a Real Setup	46
3.5 Conclusion	46

4	IPC: INCREMENTAL PROBABILISTIC CONSENSUS-BASED CONSISTENT SET MAX- IMIZATION FOR SLAM BACKENDS	49
4.1	Motivation	49
4.2	Related Work	53
4.3	Method	54
4.3.1	Assumptions and Heuristics	55
4.3.2	Problem Formulation	55
4.3.3	Independent Subgraphs and Solution Propagation	56
4.3.4	The IPC Algorithm	57
4.4	Evaluation	59
4.4.1	Datasets	60
4.4.2	Metrics	60
4.4.3	Experiments	60
4.5	Conclusions	63
5	TACO: A TEST AND CHECK FRAMEWORK FOR ROBUST POSE GRAPH OPTI- MIZATION	65
5.1	Motivation	65
5.2	Preliminaries	68
5.2.1	Pose Graph Optimization Formulation	68
5.3	Method	69
5.3.1	IPC: Incremental Probabilistic Consensus	70
5.3.2	kBL-IPC: k-Best Loops IPC	72
5.3.3	Switchable Constraints	74
5.3.4	SR-SC: Selective Randomized Switchable Constraints	75
5.4	Experiments	79
5.4.1	Metrics	79
5.4.2	Metrics for RPGO	82
5.4.3	Datasets	82
5.4.4	Effect of the Parameter K in kBL-IPC	84
5.4.5	IPC vs IPC+SR-SC	84
5.4.6	Comparison with State-of-the-art Methods	85
5.4.7	Runtime Comparison	89
5.5	Conclusions	89
6	CONCLUSIONS	91
6.1	Future Work	92

List of Figures

1.1	Examples of Autonomous Robots	2
1.2	Common Data Association Problems	4
2.1	Deterministic Evolution of a Robot's Dynamic State	10
2.2	Probabilistic Evolution of a Robot's Dynamic State	11
2.3	Dynamic Bayes Networks	12
2.4	Pose Graph Diagram	14
2.5	New Edge Representation for the Pose Graph	16
2.6	Pose Graph Formalization Using Errors	16
2.7	Effects of Constraints on Pose Graphs	25
2.8	Illustration of Manifold Operations	28
2.9	Factor Graph Formulation for Simultaneous Localization and Mapping . . .	31
2.10	Factor Graph Formulation for Tracking	31
2.11	Factor Graph Formulation for Structure-from-Motion	32
2.12	Factor Graph Formulation for Optimal Control	33
2.13	Factor Graph Formulation for the Switching Systems Problem	34
3.1	Multi-camera Eye-on-Base Calibration	36
3.2	Single-camera Eye-on-Base Calibration	39
3.3	Graphical Representation of the Optimization Problem for Eye-on-Base Single Camera Setups	40
3.4	Evolution of the Graph Representation in Multi-Camera Setups	42
3.5	Simulated Evaluation Results: Comparisons with Prior Approaches	44
3.6	Simulated Evaluation Results: Single and Multi-Camera	45
3.7	Real-world Experimental Setup	47
3.8	Real-world Experimental Setup: Rear View	47
4.1	Overview of the IPC Framework Integrated into a Visual SLAM System . . .	50
4.2	Trajectories for the FR079 Dataset	52
4.3	Example of Independent Sub-graph in a Pose Graph	57
4.4	Precision, Recall and F1 Curves for Varying Outlier Percentages	61
4.5	ATE, RPE and Convergence Time Curves for Varying Outlier Percentages . .	62
5.1	Overview of the TACO Framework	66
5.2	Example of Pose Graph with Odometry Constraints	70
5.3	Example Pose Graph Showing Intersection and Union	72

5.4	The Switchable Constraint Formulation	74
5.5	Examples of Possible Trusted Sub-graphs	77
5.6	Distortion Effects from Incorrectly Integrating Outliers	80
5.7	Examples of ATE and RPE Scores for Trajectories Under High Distortion	81
5.8	Comparison of kBL-IPC for Varying Values of k	83
5.9	Comparison of IPC, 2BL-IPC, TACO	83
5.10	Graph Showing Average Reduction in Problem Size	84
5.11	Comparison of TACO and 2BL-TACO	85
5.12	Qualitative Analysis of the Evaluated Methods on the KITTI_00 Trajectory with 100% Outlier Injection	87
5.13	Qualitative Analysis of the Evaluated Methods on the KITTI_05 Trajectory with 70% Outlier Injection	88

List of Tables

2.1	Robust Cost Functions and Their Reweighting Factors	27
3.1	Evaluation of Reprojection Errors Using the Real Setup Data	46
4.1	Summary of Metrics for 50% and 100% of Outliers	61
4.2	IPC's ACTxC for Varying Outlier Percentages	63
5.1	Comparison of Methods on 2D Datasets for Varying Percentages of Injected Outliers	86
5.2	Comparison of Methods on Visual SLAM Datasets for Varying Percentages of Injected Outliers	86
5.3	ATE and RPE Results for Methods on 2D Datasets with 50% and 100% Injected Outliers	86
5.4	ATE and RPE Results for Methods on Visual SLAM Datasets with 50% and 100% Injected Outliers	87
5.5	Comparison of Runtimes	89

Glossary

- ACTxC** Average Convergence Time per Constraint. 62, 63, 84
- ATE** Absolute Trajectory Error. 60, 62, 68, 79–82, 86–88
- CnS** Consensus Set. 57, 59, 61
- CPU** Central Processing Unit. 60, 79
- DB** Database. 66
- DBN** Dynamic Bayes Network. 12–14
- FG** Factor Graph. 30–34
- GD** Gradient Descent. 19–21, 29
- GN** Gauss-Newton. 16, 18–21, 26, 29, 69
- HE** Hand-Eye. i, 3–5, 35–39, 91
- IoU** Intersection over Union. 73
- IPC** Incremental Probabilistic Consensus. 5, 49–51, 55, 57, 59, 61–63, 65–67, 70–77, 79, 83–85, 89–92
- kBL** k-Best Loops. 66, 67, 70, 74–77, 79, 83, 84, 90
- LiDAR** Light Detection and Ranging. 1, 2, 4, 50
- LM** Levenberg-Marquardt. 17, 19–21, 26, 41, 69
- LQR** Linear Quadratic Regulator. 32
- MAP** Maximum a Posteriori. i, 2, 5, 35, 50, 51
- NLLS** Non-Linear Least Squares. i, 2, 5, 16, 17, 21, 23, 24, 26, 27, 51, 65, 66, 69
- PCG** Preconditioned Conjugate Gradient. 25

PDL Powell’s Dog Leg. 17, 20–22, 26, 29, 69

PG Pose Graph. i, 1–3, 5, 13–16, 24, 25, 30, 35, 46, 50, 65, 66, 91

PGO Pose Graph Optimization. i, 1, 4, 5, 49–51, 55, 57–60, 65–70, 80, 82, 89, 92

PnP Perspective-n-Point. 37

PR Place Recognition. 4, 5, 49, 66, 68

RGB Red Green Blue. 1

RMSE Root Mean Square Error. 80

RPE Relative Pose Error. 60, 62, 68, 79–82, 86–88

RPGO Robust Pose Graph Optimization. i, 4–6, 53, 54, 65–68, 79, 80, 82, 91, 92

RW-NLLS Re-Weighted Non-Linear Least Squares. 26, 27

SC Switchable Constraints. vi, 65–67, 70, 74–77, 79, 84, 85, 89, 90

SE Special Euclidean Group. 10, 38, 55, 68, 79

SfM Structure-from-Motion. i, 2, 31, 32

SLAM Simultaneous Localization and Mapping. i, 1–5, 9, 11, 14, 26, 27, 30, 31, 35, 36, 49, 50, 66–68, 75, 79, 80, 82–88, 91

SLDS Switching Linear Dynamic System. 32

SO Special Orthogonal Group. 28, 29

SR Selective Random. vi, 66, 67, 75–77, 79, 84, 85, 89, 90

SVD Singular Value Decomposition. 38

TACO Test And Check Optimization. 6, 65–67, 69, 88, 91

VPR Visual Place Recognition. 50, 66

1

Introduction

In the coming years, humans and robots will collaborate on a daily basis to accomplish shared tasks. This synergy is made possible by the advent of large generalist models [1, 2], which enable robots to understand the sequence of actions necessary to fulfill the high-level tasks requested by humans. A high-level task consists of multiple lower-level sub-tasks that must be executed in the correct order to ensure successful completion. To perform tasks effectively in real-world environments, a robot must possess the capability to both perceive and understand its surroundings [3]. This is achieved through the integration of various sensors that provide the robot with essential information about its environment. Among the most widely used sensors are RGB cameras, which mimic human vision by capturing visual information, and LiDARs, which allow the robot to perceive the geometry and spatial structure of its surroundings. These sensors not only enable robots to measure their own movement, but also facilitate the construction of an internal representation, or map, of the environment [4], which plays a crucial role in enabling a robot to successfully complete a task in an autonomous way. For these sensors to function optimally, careful intrinsic and extrinsic calibration of the entire sensor suite is necessary. The aforementioned problems provide a number of fundamental questions that should be carefully addressed:

- *What about my sensors?* (Calibration)
- *Where am I?* (Localization)
- *What is happening around me?* (Scene Understanding)
- *What's around me?* (Mapping)
- *How can I reach the goal?* (Path Planning and Navigation)

This thesis investigates *robust state estimation* [5], with a particular focus on enhancing SLAM and Calibration through Pose Graph Optimization (PGO). PGs provide a powerful frame-

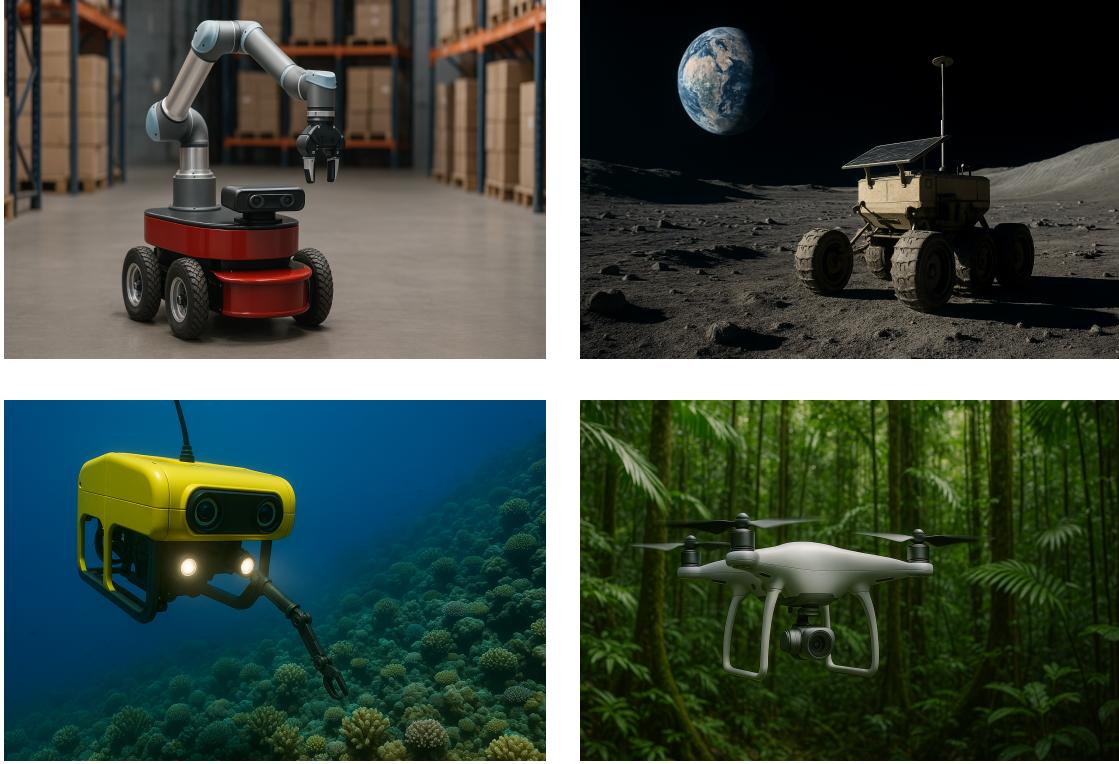


Figure 1.1: Images showcasing various types of autonomous robots across different environments.

work for formulating a broad range of robotics problems as Maximum a Posteriori (MAP) estimation problems, which can be efficiently solved using Non-Linear Least Squares (NLLS) optimization. The solution of an NLLS problem not only yields the best estimate for the MAP problem but also provides the associated uncertainty, enabling the system to identify challenging scenarios and trigger recovery procedures when necessary. The flexibility of PGs allows them to model a wide range of problems, including Structure-from-Motion (SfM) [6, 7], Optimal Control [8, 9], Extrinsic Calibration [10, 11], Tracking, Point Cloud Registration [12, 13], Visual and LiDAR Odometry [14, 15], Message Passing [16, 17], and SLAM [18, 19].

Beyond their generality, PGs inherently incorporate uncertainty, making them well-suited for handling noisy sensor measurements—provided that the noise models are appropriately defined. However, to ensure correct optimization, two key assumptions must hold [20]:

- A good initialization of the state variables is provided.
- The *data association* problem [21] is correctly solved, ensuring a correct correspondence between the measurements and the state variables (Fig. 1.2).

Historically, robotic applications were primarily confined to controlled environments where noise sources were minimal and carefully managed. In such settings, failures had limited consequences, and if an issue arose, data collection could simply be repeated, or manual corrections

could be applied by a human operator. Consequently, research predominantly focused on ensuring that robots could produce estimates as *accurately* as possible. However, as robotics expands into real-world applications, the emphasis has shifted towards *robustness* [22]. In this context, robustness is defined as the ability to consistently produce reliable results that enable the correct execution of tasks, even in challenging and unpredictable conditions. This capability is crucial for deploying robots in open-world environments, where various dynamic and uncertain factors can degrade system performance. Achieving robustness is a key enabler for the widespread adoption of robots in practical applications. Robustness is particularly critical because the sources of *noise* vary depending on the type of sensor and the environment in which the robot operates. For perception algorithms relying on vision, common sources of noise include motion blur, varying lighting conditions, and changes in appearance (e.g., seasonal variations). These challenges are present in most environments and can severely affect the performance of vision-based perception systems. In addition to these general issues, specific environments introduce unique challenges. In underwater environments, for example, cameras are affected by light absorption, refraction, turbidity, and unstructured featureless scenes. In this thesis, we address the challenge of enhancing *robustness* specifically in two key problems: Hand-Eye calibration and SLAM.

Hand-Eye Calibration: The Hand-Eye (HE) calibration is a specific instance of extrinsic calibration problems [23]. The HE is defined as the problem of estimating the spatial transformation between a robot reference frame (e.g., robot base or end-effector frames) and the reference frame of the camera. Accurate knowledge of these transformations is crucial since visual information captured by the camera is expressed within its own reference frame. Therefore, for the robot to effectively act upon this visual information, the transformation between the robot and camera frames must be precisely known. Typically, visual noise represents the most significant source of uncertainty during the HE calibration procedure, as robotic arm kinematics are usually well-calibrated by the manufacturer. Visual noise manifests in various forms, such as motion blur, variations in illumination, and miss-detections. Failure to account for these factors during the calibration process can result in incorrect estimations, often necessitating tedious trial-and-error repetitions to achieve accurate results. Given their inherent capability to explicitly model noise within the optimization process, PGs serve as an ideal tool for addressing these challenges, substantially reducing human intervention typically required for tasks such as data recollection, noise identification, and noise removal from calibration setups.

Simultaneous Localization and Mapping: By constructing a structured representation of the environment, the robot can not only determine its own position but also identify objects of interest and evaluate the feasible paths to reach them or accomplish a given objective. The process of grounding perception to a structured 3D representation of the world is known as *spatial perception*. Among the various challenges in spatial perception, this thesis focuses on the Simultaneous Localization and Mapping (SLAM) problem [18, 19]. SLAM involves the

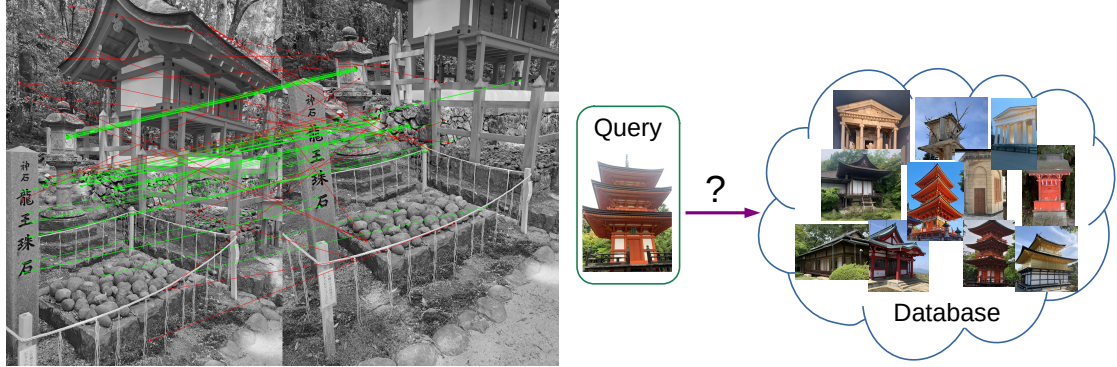


Figure 1.2: Two different types of data association problems are illustrated. On the left, the goal is to match the common features between the two images, with correct matches highlighted in green and incorrect ones in red. On the right, the goal is to match the query image with the corresponding image in the database that depicts the same location as in the query.

incremental construction of a map of an unknown environment while simultaneously localizing the robot within the same map. Successfully solving the SLAM problem allows access to accurate geometric-semantic representations of the environment [24, 25] and provides a refined estimate of the robot’s trajectory during exploration missions.

A common and critical data association problem in SLAM is Place Recognition (PR) [26, 27], which involves recognizing when a robot has returned to a previously visited location. This requires associating sensor measurements—such as images or LiDAR scans—with the correct state variables representing that location. However, due to the inherent difficulty of data association, errors frequently occur, and these mistakes—referred to as *outliers*—can have catastrophic effects on the optimization process. One of the main challenges researchers aim to solve, to enable autonomous robotic exploration in hazardous, GPS-denied environments, is developing Robust Pose Graph Optimization (RPGO) [28] frameworks that are resilient to outliers. Addressing this issue is the central focus of this thesis.

Both the Calibration and SLAM problems represent essential technologies for enabling autonomous capabilities in mobile robots. A prominent example includes mobile manipulators, such as the one depicted in Fig. 1.1, which necessitate precise HE calibration for accurate object manipulation and robust SLAM solutions to effectively navigate toward objectives in unstructured environments.

1.1 MAIN CONTRIBUTIONS

The first key contribution of this thesis is demonstrating the effectiveness of Pose Graph Optimization (PGO) as a foundational tool for developing robust spatial perception algorithms.

We showcase its flexibility and general applicability, by illustrating how PGO can be leveraged to solve the HE Calibration problem. This example is one of many complex estimation problems encountered across various robotic perception tasks. The second, and most significant, contribution of this work is addressing one of the most critical challenges in Simultaneous Localization and Mapping (SLAM): the failure of Place Recognition (PR), which leads to incorrect loop closures and outlier measurements being introduced into the PG. This issue is particularly problematic in environments with strong perceptual aliasing, such as underwater or cave exploration, where visually or geometrically similar locations can result in erroneous data associations. Since PGO is highly sensitive to outliers, even a small number of incorrect constraints can significantly degrade the estimated trajectory and, in extreme cases, cause complete failure of the optimization process. To overcome this fundamental limitation, we introduce two novel Robust Pose Graph Optimization (RPGO) frameworks designed to enhance robustness against outliers. Our approach builds upon state-of-the-art techniques in robust estimation, leveraging strategies such as consistency checks, adaptive weighting schemes, and multi-horizon reasoning to identify and mitigate the impact of outliers during autonomous operations. By improving the resilience of PGO in highly ambiguous environments, our proposed methods provide a more reliable foundation for long-term autonomy. In this section, we provide a detailed overview of the structure and organization of the thesis, outlining the key chapters and their respective contributions.

In Chapter 2, we present how the evolution of a robot’s state can be described as a MAP estimation problem, providing a formal framework for state estimation. Next, we establish the connection between the NLLS optimization approach and the MAP problem under Gaussian noise assumptions, illustrating the mathematical foundations that enable this formulation. Finally, we demonstrate how an NLLS problem can be structured using a PG graphical model, showcasing its flexibility by expressing a variety of example problems within this framework.

In Chapter 3, we introduce a PG formulation for Hand-Eye (HE) calibration, providing a unified framework that enhances the robustness of the calibration process. By leveraging the PG approach, we effectively embed resilience to visual noise, a challenge that traditional calibration methods struggle to address, thereby enabling a more reliable HE calibration for both single and multi-camera setups. Furthermore, the inherent expressiveness and flexibility of the PG framework facilitate seamless extensions to accommodate various camera models, making it a versatile tool for different calibration scenarios. Chapter 3 serves as a compelling demonstration of how the powerful PG methodology can be harnessed to develop robust and adaptable spatial perception algorithms.

In Chapter 4, we introduce the first version of our proposed Robust Pose Graph Optimization (RPGO) framework, designed to enhance SLAM performance in highly aliased environments such as underwater or cave exploration scenarios. At the core of this framework lies a greedy algorithm, namely IPC, which is founded on the principle of *consistency* between measurements, and different key insights from various state-of-the-art research contributions. Through extensive evaluation on a diverse set of 2D datasets, IPC demonstrates superior re-

silience to outliers compared to existing methods, making it a reliable solution for challenging real-world applications.

In Chapter 5, we address the limitations of IPC by introducing an extended framework for RPGO, called TACO. TACO operates on two distinct time horizons, ensuring online performance by making rapid decisions in the short term at the cost of reduced accuracy, while simultaneously performing more thorough error corrections over a longer horizon. Furthermore, this chapter includes a discussion on the most meaningful metrics for evaluating RPGO, providing insights into how different evaluation criteria impact the assessment of the robustness of different frameworks for RPGO.

In Chapter 6, we summarize the conclusions drawn from this research journey. Additionally, we discuss the limitations and potential future research directions that build upon this thesis, exploring how our methodologies can be extended, refined, or applied to new and more complex scenarios in the field.

1.2 LIST OF PUBLICATIONS

The following peer-reviewed conference and journal articles contributed to this thesis:

- D. Evangelista, **E. Olivastri**, D. Allegro, E. Menegatti and A. Pretto, “A Graph-Based Optimization Framework for Hand-Eye Calibration for Multi-Camera Setups”. 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom.
- **E. Olivastri** and A. Pretto, “IPC: Incremental Probabilistic Consensus-based Consistent Set Maximization for SLAM Backends”. 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan.
- **E. Olivastri**, and A. Pretto, “TACO: A Test and Check Framework for Robust Pose Graph Optimization”. IEEE Transactions on Robotics (T-RO) [Submitted].

The following list of peer-reviewed conference and journal publications, while not included in this thesis, are related to the topic of *robust state estimation*, with applications in autonomous driving, underwater localization, and dynamic mapping, but were not included in core chapters of this thesis:

- **E. Olivastri**, J. Francis, A. Pretto, N. Suenderhauf, and K. Rana, “Multi-Modal 3D Scene Graph Updater for Shared and Dynamic Environments”. 2024 Workshop on Lifelong Learning for Home Robots at the Conference on Robot Learning (CoRL), Munich, Germany.
- **E. Olivastri**, D. Fusaro, L. Wanmeng, S. Mosco and A. Pretto, “A Sonar-based AUV Positioning System for Underwater Environments with Low Infrastructure Density”.

2024 Workshop on Field Robotics at IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan.

- A. Bacchin, L. Barcellona, S. Shamsizadeh, **E. Olivastri**, A. Pretto and E. Menegatti, “PanNote: an Automatic Tool for Panoramic Image Annotation of People’s Positions”. 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan.
- I. Donadi, **E. Olivastri**, D. Fusaro, L. Wanmeng, D. Evangelista and A. Pretto, “Improving Generalization of Synthetically Trained Sonar Image Descriptors for Underwater Place Recognition”. 2023 International Conference on Computer Vision Systems (ICVS), Vienna, Austria.
- D. Fusaro, **E. Olivastri**, D. Evangelista, M. Imperoli, E. Menegatti and A. Pretto, “Pushing the Limits of Learning-based Traversability Analysis for Autonomous Driving on CPU”. 2023 Intelligent Autonomous Systems (IAS), Zagreb, Croatia.
- D. Fusaro, **E. Olivastri**, D. Evangelista and A. Pretto, “An Hybrid Approach to Improve the Performance of Encoder-Decoder Architectures for Traversability Analysis in Urban Environments”. 2022 IEEE Intelligent Vehicles Symposium (IV), Aachen, Germany.
- D. Fusaro, **E. Olivastri**, I. Donadi, D. Evangelista, E. Menegatti and A. Pretto, “Pyramidal 3D Feature Fusion on Polar Grids for Fast and Robust Traversability Analysis on CPU”. 2023 IEEE Robotics and Autonomous Systems (RAS).
- **E. Olivastri** and A. Pretto, “Towards Outliers-Resistant Optimization for SLAM”. 2023 Italian Conference for Robotics and Intelligent Machines (I-RIM), Roma, Italy.
- L. Wanmeng, **E. Olivastri**, D. Fusaro, D. Evangelista, E. Menegatti and A. Pretto, “Robust Sonar Image Descriptors for Underwater Place Recognition Trained in Synthetic Scenarios”. 2023 Italian Conference for Robotics and Intelligent Machines (I-RIM), Roma, Italy.
- D. Evangelista, I. Donadi, D. Fusaro, **E. Olivastri** and A. Pretto, “Towards Accurate 3D Positioning in Large-Scale Underwater Environments”. 2022 Italian Conference for Robotics and Intelligent Machines (I-RIM), Milano, Italy.

1.3 OPEN-SOURCE CONTRIBUTIONS

The implementations of the algorithms presented in this thesis have been made open-source to facilitate straightforward reproducibility of results and enable comparisons with future algorithms:

- The implementation for “A Graph-Based Optimization Framework for Hand-Eye Calibration for Multi-Camera Setups” presented in Chapter 3 is available at:
https://bitbucket.org/freelists/gm_handeye
- The implementation for “IPC: Incremental Probabilistic Consensus-based Consistent Set Maximization for SLAM Backends” presented in Chapter 4 is available at:
<https://github.com/EmilioOlivastri/IPC>
- The implementation for “TACO: A Test and Check Framework for Robust Pose Graph Optimization” presented in Chapter 5 is available at:
<https://github.com/EmilioOlivastri/TACO>

2

Preliminaries

Robotics revolves around state estimation [29]. Robots perceive the external environment through sensors, as inferring both their own state and the surrounding world is essential for completing tasks.

But what is a state? A state can be thought of as the set of all information about the robot and the environment that is required to complete a specific task, meaning that the state is a task-dependent concept. States can be classified into two categories: *dynamic* and *static*. Dynamic states evolve and change over time, while static states remain unaltered. In this thesis, state variables will typically be denoted by the symbol x_t , where t indicates the value of the variable at a specific time. Bold notation will be used to indicate vectors, while bold capital letters will denote matrices.

Here, we focus on Simultaneous Localization and Mapping [30, 18, 19], which involves simultaneously estimating a map of an unknown environment while localizing within it. When solving SLAM, the state that typically needs to be estimated can be:

- the current pose of the robot $\mathbf{x}_t$;
- the full trajectory of the robot $\{\mathbf{x}_0, \dots, \mathbf{x}_t\}$;
- the current pose of the robot and the map of the environment $\{\mathbf{x}_t, \mathcal{M}\}$;
- the full trajectory of the robot and the map of the environment $\{\mathbf{x}_0, \dots, \mathbf{x}_t, \mathcal{M}\}$;

To understand how it is possible to estimate the robot's state at different times, it is important to first understand how a dynamic system evolves. Fig. 2.1 illustrates the evolution of the robot's dynamic variables. The function f , commonly known as the transition function, describes the evolution of the state from x_{t-1} to x_t , given the control input u_{t-1} . The function h , typically referred to as the observation function, relates the robot's current state to an observation z_t

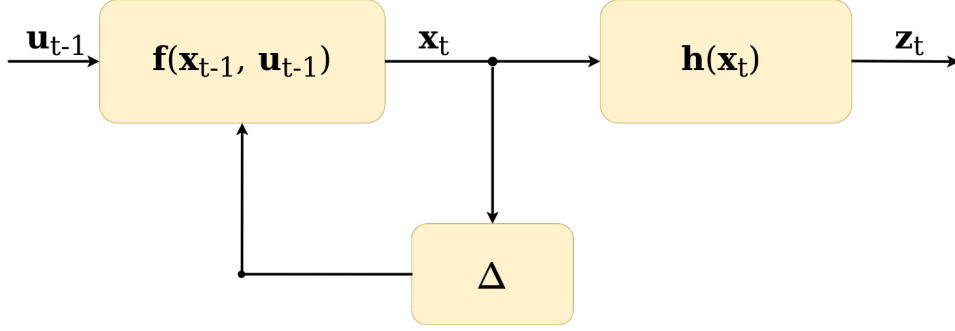


Figure 2.1: Deterministic evolution of a robot's dynamic state. The control input u_{t-1} , together with the previous state x_{t-1} , is used to estimate the current state x_t through the transition function $f(\cdot)$. Finally, the state and observation are related through the observation function $h(\cdot)$.

of the external world. To make these concepts more concrete, we provide an example of how these different elements appear in the case of a wheeled robot moving in 2D space:

- $\mathbf{x}_t = [x_t, y_t, \theta_t] \in \mathbb{R}^2 \times \mathbb{S}$ represents the state of the robot, where we aim to estimate only its position and orientation;
- $\mathbf{u}_t = [v, w] \in \mathbb{R}^2$ denotes the velocity control, where the desired translational and rotational velocities are provided as inputs to the robot;

The transition function for $\mathbf{x}_t \in \text{SE}(2)$, assuming velocity control with a constant velocity model, is given by:

$$\mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{v_{t-1}}{w_{t-1}} \sin(\theta_{t-1} + w_{t-1} \Delta t) \\ -\frac{v_{t-1}}{w_{t-1}} \cos(\theta_{t-1} + w_{t-1} \Delta t) \\ w_{t-1} \Delta t \end{bmatrix}, \quad (2.1)$$

where Δt is the time elapsed between $t-1$ and t , and ideally, $\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. Assuming the robot can always perceive a known and static landmark $\mathbf{l} = [x_l, y_l]$, a possible observation function $\mathbf{h}$ could be:

$$\mathbf{h}(\mathbf{x}_t) = \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) \\ \sin(\theta_t) & \cos(\theta_t) \end{bmatrix} \begin{bmatrix} x_l - x_t \\ y_l - y_t \end{bmatrix} \quad (2.2)$$

As the observation $\mathbf{z}_t = [\hat{x}_l, \hat{y}_l]$ relative to the landmark $\mathbf{l}$ is expressed in the robot's reference frame, the observation function transforms the known location of the landmark $\mathbf{l}$ from the world reference frame to the robot's reference frame.

In an ideal system, both $\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t)$ and $\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. However, in reality, neither the control input nor the observation is perfect, as both sensors and actuators are affected by a certain amount of noise. A fundamental assumption regarding the noise affecting the robot is

that it follows a zero-mean Gaussian distribution, $\mathbf{n} \approx \mathcal{N}(\mathbf{0}, \Sigma)$. Consequently, the system's evolution is better represented by the schema in Fig. 2.1, where the transition and observation models become $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ and $p(\mathbf{z}_t | \mathbf{x}_t)$, respectively.

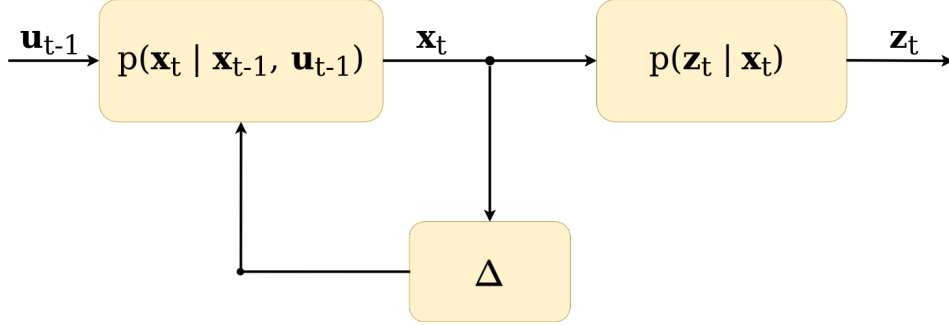


Figure 2.2: Probabilistic evolution of a robot's dynamic state. The transition and observation models become, respectively, $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ and $p(\mathbf{z}_t | \mathbf{x}_t)$. This formulation accounts for the noise affecting different sensors and model approximations.

2.1 DYNAMIC BAYESIAN NETWORK MODEL

From a probabilistic perspective, solving the SLAM problem means estimating the set of variables $\mathbf{x}_{1:t}$ that maximize the posterior of the robot's state, given the robot's initial state $\mathbf{x}_0$, all measurements, and input commands.

$$\mathbf{x}_{1:t}^* = \underset{\mathbf{x}_{1:t}}{\operatorname{argmax}} p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) \quad (2.3)$$

The first step is to manipulate Eq. (2.3) by applying the Bayes' rule:

$$p(\mathbf{x}_{1:t} | \mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) = \frac{p(\mathbf{z}_t | \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0)}{p(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0)} \quad (2.4)$$

The denominator can be considered a normalization factor, as it does not incorporate any of the state variables of interest:

$$\eta_t = \frac{1}{p(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0)} \quad (2.5)$$

so Eq. (2.4) becomes:

$$p(\mathbf{x}_{1:t} | \mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) = \eta_t p(\mathbf{z}_t | \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) \quad (2.6)$$

From Fig. 2.3, various considerations can be made to further simplify the above equation. The figure presents a graphical model, known as a Dynamic Bayes Network (DBN) [31], which illustrates the system's evolution, its state variables, and their corresponding control inputs and observations. From this schema, we can also recognize a Markov Chain [32], which allows us to exploit the *conditional independence* between variables, derived from the Markov assumptions. Consequently, the first term of Eq. (2.6) simplifies to:

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) = p(\mathbf{z}_t \mid \mathbf{x}_t) \quad (2.7)$$

Given the variable $\mathbf{x}_t$, the information from previous states, controls, and measurements does not add any new information, as $\mathbf{z}_t$ depends solely on $\mathbf{x}_t$. For the second term instead, $\mathbf{x}_t$ is

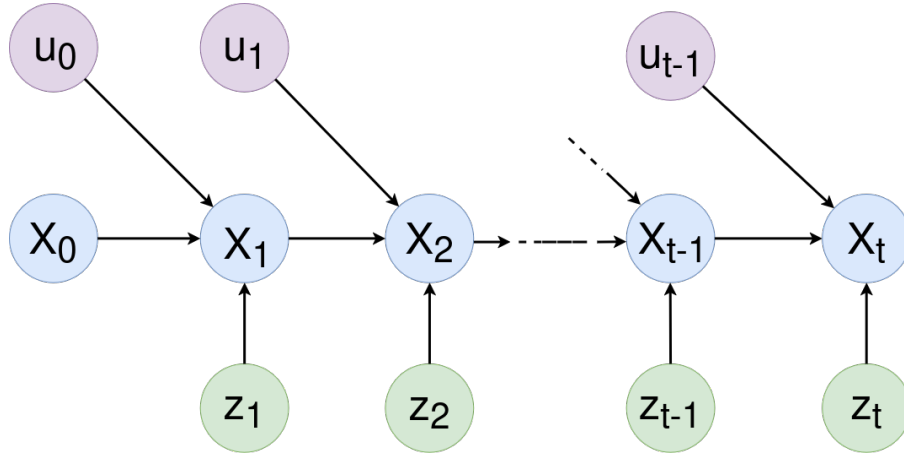


Figure 2.3: Dynamic Bayes Network (DBN) models the evolution of a robot's state $\mathbf{x}_{1:t}$, given the control inputs $\mathbf{u}_{0:t-1}$ and the observations $\mathbf{z}_{1:t}$.

separated from $\mathbf{x}_{1:t}$, and then the chain rule is applied:

$$p(\mathbf{x}_t, \mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) = p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) \quad (2.8)$$

The first term of Eq. (2.8) can be further simplified using the same considerations as done for Eq. (2.7). From Fig. 2.3, it is clear that given $\mathbf{x}_{t-1}$ and $\mathbf{u}_{t-1}$, all other observations and controls become redundant for estimating the probability of $\mathbf{x}_{t-1}$:

$$p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \quad (2.9)$$

Finally, using Eq. (2.7), Eq. (2.8), and Eq. (2.9) in Eq. (2.6) we obtain:

$$p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) = \eta_t p(\mathbf{z}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) \quad (2.10)$$

where the last term is the same as Eq. (2.3), but at the previous time step. The temporal structure given by the DBN can be exploited to further simplify Eq. (2.10):

$$p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{x}_0) = p(\mathbf{x}_0) \prod_{k=1}^t \eta_k p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \quad (2.11)$$

In Eq. (2.11), we can recognize two main terms that we encountered previously: the observation model $p(\mathbf{z}_k \mid \mathbf{x}_k)$, and the motion model $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1})$. The equation also highlights the temporal structure of the DBN, as all the probabilities relative to the previous states $\{\mathbf{x}_1, \dots, \mathbf{x}_{k-1}\}$ must be computed to estimate the ones relative to the variable $\mathbf{x}_k$. The sequential nature of Eq. (2.11) is powerful when computing the latest estimates, as all the products have been computed already up to the step $t - 1$. However, this comes with a downside: if an error is discovered at a previous step j , all the estimates need to be updated sequentially from that index, and the operation cannot be parallelized. Finally, solving Eq. (2.3) is equivalent to:

$$\mathbf{x}_{1:t}^* = \underset{\mathbf{x}_{1:t}}{\operatorname{argmax}} \prod_{k=1}^t p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \quad (2.12)$$

Both η_k and $p(\mathbf{x}_0)$ are not considered here, as they don't affect the solution.

2.2 POSE GRAPH MODEL

To remove the temporal dependency, the DBN structure can be modified into another graphical representation called Pose Graph (PG) [33]. The PG instead highlights the spatial structure of the problem, as depicted in Fig. 2.4: Here we consider the control inputs $\mathbf{u}_{1:t}$ as other observations, and the motion model can also be integrated into the observation model as *observed motion*, so Eq. (2.4) can be simplified into:

$$\mathbf{x}_{1:t}^* = \underset{\mathbf{x}_{1:t}}{\operatorname{argmax}} p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{x}_0) \quad (2.13)$$

Here, the Bayes rule can be easily applied to $p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{x}_0)$ to obtain:

$$p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{x}_0) = \frac{p(\mathbf{z}_{1:t} \mid \mathbf{x}_{0:t}) p(\mathbf{x}_{1:t})}{p(\mathbf{z}_{1:t})} \propto p(\mathbf{z}_{1:t} \mid \mathbf{x}_{0:t}) \quad (2.14)$$

As $p(\mathbf{z}_{1:t})$ is a constant for Eq. (2.13), it does not impact its solution. The prior $p(\mathbf{x}_{1:t})$ is considered uniformly distributed, as we don't possess any prior information on the state $\mathbf{x}_{1:t}$, therefore not impacting Eq. (2.13) either. Thus, to estimate the value of $\mathbf{x}_{1:t}$ that maximize the posterior in Eq. (2.14), only the observation model $p(\mathbf{z}_{1:t} \mid \mathbf{x}_{0:t})$ needs to be considered.

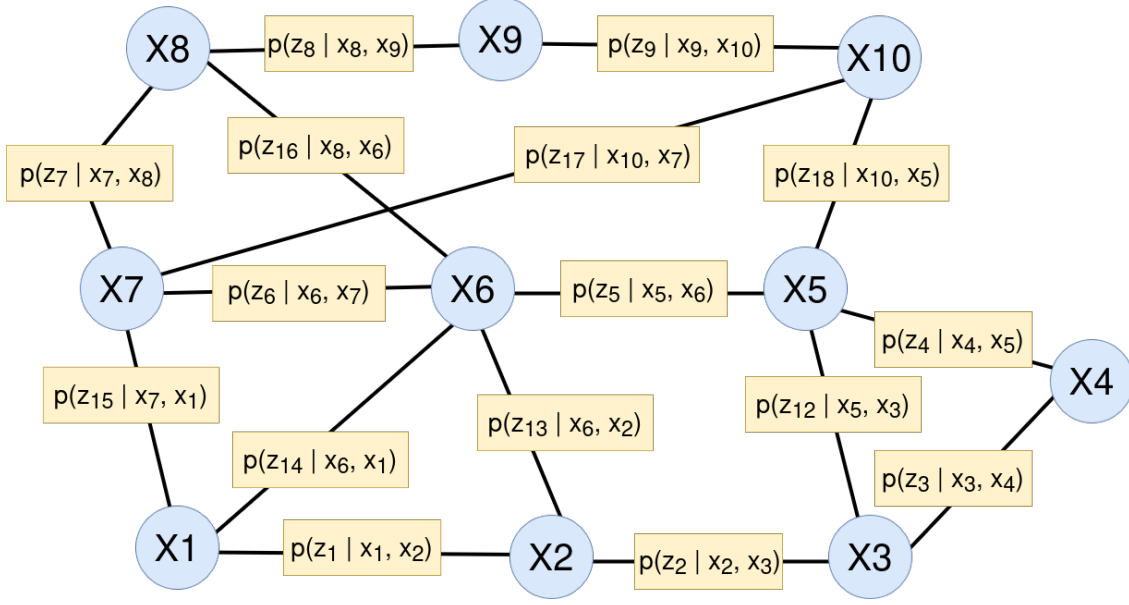


Figure 2.4: An alternative graphical model for representing the evolution of the state $\mathbf{x}_{1:t}$. In the Pose Graph (PG), each edge captures the dependency between variables and associates it with the corresponding observation $\mathbf{z}_k$. When state variables have consecutive indices $[i, i + 1]$, the observation encodes the motion between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. This is referred to as *observed motion*, which now replaces the control inputs $\mathbf{u}_{0:t-1}$ previously used in the DBN.

From Fig. 2.4, similar considerations on *conditional independence* can be made: given the variables they depend on, the observations become independent from one another, allowing us to rewrite Eq. (2.14) in the following way:

$$p(\mathbf{z}_{1:t} | \mathbf{x}_{0:t}) = \prod_{k=1}^t p(\mathbf{z}_k | \mathbf{x}_{0:t}) \quad (2.15)$$

As the $\mathbf{z}_k$, in the PG case, depends only on $\mathbf{x}_i$ and $\mathbf{x}_j$, the other state variables don't impact the posterior of $\mathbf{z}_k$:

$$p(\mathbf{z}_k | \mathbf{x}_{0:t}) = p(\mathbf{z}_k | \mathbf{x}_i, \mathbf{x}_j) \quad (2.16)$$

The optimal estimate of the state variables $\mathbf{x}_{1:t}$ for PG SLAM is the following:

$$\mathbf{x}_{1:t}^* = \underset{\mathbf{x}_{1:t}}{\operatorname{argmax}} \prod_{k=1}^t p(\mathbf{z}_k | \mathbf{x}_i, \mathbf{x}_j) \quad (2.17)$$

As mentioned previously, $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{n}_{z_k}$, where $\mathbf{n}_{z_k} \approx \mathcal{N}(0, \Sigma_{\mathbf{z}_k})$ is the zero-mean Gaussian noise from the observation. Thus, the probability $p(\mathbf{z}_k | \mathbf{x}_i, \mathbf{x}_j)$ can be thought of as

the error between the measured observation $\mathbf{z}_k$ and the expected observation $\mathbf{h}(\mathbf{x}_i, \mathbf{x}_j)$, which is also Gaussian distributed:

$$p(\mathbf{z}_k | \mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\sqrt{2\pi\|\Sigma_{\mathbf{z}_k}\|}} \exp \left\{ -\frac{1}{2}(\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k)^T \Sigma_{\mathbf{z}_k}^{-1} (\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k) \right\} \quad (2.18)$$

Exploiting the Gaussian form of $p(\mathbf{z}_k | \mathbf{x}_i, \mathbf{x}_j)$ allows us to simplify Eq. (2.17):

$$\begin{aligned} \mathbf{x}_{1:t}^* &= \underset{\mathbf{x}_{1:t}}{\operatorname{argmax}} \left[\prod_{k=1}^t p(\mathbf{z}_k | \mathbf{x}_i, \mathbf{x}_j) \right] \\ &= \underset{\mathbf{x}_{1:t}}{\operatorname{argmax}} \left[\prod_{k=1}^t \frac{1}{\sqrt{2\pi\|\Sigma_{\mathbf{z}_k}\|}} \exp \left\{ -\frac{1}{2}(\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k)^T \Sigma_{\mathbf{z}_k}^{-1} (\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k) \right\} \right] \\ &= \underset{\mathbf{x}_{1:t}}{\operatorname{argmin}} \left[-\log \prod_{k=1}^t \frac{1}{\sqrt{2\pi\|\Sigma_{\mathbf{z}_k}\|}} \exp \left\{ -\frac{1}{2}(\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k)^T \Sigma_{\mathbf{z}_k}^{-1} (\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k) \right\} \right] \\ &= \underset{\mathbf{x}_{1:t}}{\operatorname{argmin}} \left[\sum_{k=1}^t -\log \left(\frac{1}{\sqrt{2\pi\|\Sigma_{\mathbf{z}_k}\|}} \right) - \sum_{k=1}^t \left(-\frac{1}{2}(\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k)^T \Sigma_{\mathbf{z}_k}^{-1} (\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k) \right) \right] \\ &= \underset{\mathbf{x}_{1:t}}{\operatorname{argmin}} \left[\sum_{k=1}^t (\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k)^T \Sigma_{\mathbf{z}_k}^{-1} (\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k) \right] \\ &= \underset{\mathbf{x}_{1:t}}{\operatorname{argmin}} \left[\sum_{k=1}^t \mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)^T \boldsymbol{\Omega}_{\mathbf{z}_k} \mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j) \right] \\ &= \underset{\mathbf{x}_{1:t}}{\operatorname{argmin}} \left[\sum_{k=1}^t \|\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)\|_{\boldsymbol{\Omega}_{\mathbf{z}_k}}^2 \right] \end{aligned} \quad (2.19)$$

where $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k$ is the error between the expected measurements $\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j)$ and the observed measurement $\mathbf{z}_k$. The information matrix $\boldsymbol{\Omega}_{\mathbf{z}_k}$ is the inverse of the covariance matrix $\Sigma_{\mathbf{z}_k}$ relative to the measurement $\mathbf{z}_k$, and $\|\cdot\|_{\boldsymbol{\Omega}}^2$ is referred to as the squared Mahalanobis distance [34].

Now the edges between the different nodes have taken a different meaning. With Eq. (2.19), we can update Fig. 2.4 to obtain a more formal definition for the PG formulation in Fig. 2.6. The goal is then to find the estimates of the state variables $\mathbf{x}_{1:t}$ that minimize the cost function that represents the negative log likelihood of all the measurements $\mathbf{z}_{1:t}$:

$$F(\mathbf{x}_{1:t}) = \sum_{k=1}^t \|\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)\|_{\boldsymbol{\Omega}_{\mathbf{z}_k}}^2 \quad (2.20)$$

Levenberg-Marquardt (LM) [37], and Powell's Dog Leg (PDL) [38] algorithms. To simplify the notation for the derivation of the different optimization algorithms for NLLS, now $\mathbf{x}_{1:t} = \mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_k, \dots, \mathbf{x}_t]$. These strategies operate in an iterative manner:

- Compute the perturbation step $\Delta \mathbf{x}$ that minimizes the quadratic approximation of $F(\mathbf{x})$;
- Apply the perturbation $\Delta \mathbf{x}$ to the current estimate of the state variables $\hat{\mathbf{x}}$;
- Check if a local optimum has been reached; if not, repeat the procedure.

As $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)$ is non-linear, we use its first-order Taylor expansion around the current estimate $\hat{\mathbf{x}}$ as a linear approximation:

$$\begin{aligned}
 \mathbf{e}_k(\hat{\mathbf{x}}_i + \Delta \mathbf{x}_i, \hat{\mathbf{x}}_j + \Delta \mathbf{x}_j) &= \mathbf{e}_{kij}(\hat{\mathbf{x}} + \Delta \mathbf{x}) \\
 &\approx \mathbf{e}_{kij}(\hat{\mathbf{x}}) + \left. \frac{\partial \mathbf{e}_{kij}(\hat{\mathbf{x}} + \Delta \mathbf{x})}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=0} \Delta \mathbf{x} \\
 &\approx \mathbf{e}_{kij}(\hat{\mathbf{x}}) + \left. \frac{\partial [\mathbf{h}_{kij}(\hat{\mathbf{x}} + \Delta \mathbf{x}) - \mathbf{z}_k]}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=0} \Delta \mathbf{x} \quad (2.21) \\
 &\approx \mathbf{e}_{kij}(\hat{\mathbf{x}}) + \underbrace{\left. \frac{\partial \mathbf{h}_{kij}(\hat{\mathbf{x}} + \Delta \mathbf{x})}{\partial \Delta \mathbf{x}} \right|_{\Delta \mathbf{x}=0}}_{\mathbf{J}_{kij}} \Delta \mathbf{x} \\
 &\approx \mathbf{e}_{kij}(\hat{\mathbf{x}}) + \mathbf{J}_{kij} \Delta \mathbf{x}
 \end{aligned}$$

We can use the approximation in Eq. (2.21) to compute the quadratic approximation of the function $F(\mathbf{x})$, called $L(\mathbf{x})$, in the following manner:

$$\begin{aligned}
 F(\hat{\mathbf{x}} + \Delta \mathbf{x}) &= \sum_{k=1}^t \|\mathbf{e}_{kij}(\hat{\mathbf{x}} + \Delta \mathbf{x})\|_{\Omega_{\mathbf{z}_k}}^2 \\
 &\approx \sum_{k=1}^t \|\mathbf{e}_{kij}(\hat{\mathbf{x}}) + \mathbf{J}_{kij} \Delta \mathbf{x}\|_{\Omega_{\mathbf{z}_k}}^2 \\
 &= \sum_{k=1}^t \underbrace{\mathbf{e}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{e}_{kij}}_{c_{kij}} + 2 \underbrace{\mathbf{e}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kij}}_{\mathbf{b}_{kij}} \Delta \mathbf{x} + \Delta \mathbf{x}^T \underbrace{\mathbf{J}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kij}}_{\mathbf{H}_{kij}} \Delta \mathbf{x} \quad (2.22) \\
 &= \sum_{k=1}^t c_{kij} + 2 \mathbf{b}_{kij} \Delta \mathbf{x} + \Delta \mathbf{x}^T \mathbf{H}_{kij} \Delta \mathbf{x} \\
 L(\Delta \mathbf{x})_{\hat{\mathbf{x}}} &= c + 2 \mathbf{b} \Delta \mathbf{x} + \Delta \mathbf{x}^T \mathbf{H} \Delta \mathbf{x}
 \end{aligned}$$

Gauss Newton: The Gauss-Newton (GN) [36] method's goal is to estimate the $\Delta \mathbf{x}$ that minimizes the quadratic approximation $L(\Delta \mathbf{x})_{\hat{\mathbf{x}}}$ in Eq. (2.22), which is obtained by solving the linear system:

$$\begin{aligned} \frac{\partial L(\Delta \mathbf{x})_{\hat{\mathbf{x}}}}{\partial \Delta \mathbf{x}} &= 0 \\ \frac{\partial [c + 2\mathbf{b}\Delta \mathbf{x} + \Delta \mathbf{x}^T \mathbf{H} \Delta \mathbf{x}]}{\partial \Delta \mathbf{x}} &= 0 \\ 2\mathbf{b} + 2\mathbf{H}\Delta \mathbf{x} &= 0 \end{aligned} \quad (2.23)$$

which reduces to solving the linear system:

$$\mathbf{H}\Delta \mathbf{x} = -\mathbf{b} \quad (2.24)$$

The solution of Eq. (2.24) $\Delta \mathbf{x}^{gn}$ is then used to refine the current estimate $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = \hat{\mathbf{x}} + \Delta \mathbf{x}^{gn} \quad (2.25)$$

If the quadratic approximation $L(\Delta \mathbf{x})_{\hat{\mathbf{x}}}$ provides a good fit around the linearization point $\hat{\mathbf{x}}$ for the original cost function $F(\mathbf{x})$, and if a good initial estimate is provided, the GN method converges quadratically. However, if the first-order Taylor expansion does not provide a sufficiently accurate fit, $\Delta \mathbf{x}^{gn}$ may move away from the optimal solution.

Algorithm 1: Gauss-Newton

```

1 Input:  $\hat{\mathbf{x}}, \mathcal{C} = \{(\mathbf{z}_k, \Omega_{\mathbf{z}_k})\}$  // Initial Guess and Constraints
2 while !converged do
3    $\mathbf{H}, \mathbf{b} \leftarrow \mathbf{0}$ 
4   for all  $k \in \{0, \dots, t\}$  do
5      $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j) \leftarrow \mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k$ 
6      $\mathbf{J}_{ki} \leftarrow \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_i} \Big|_{\mathbf{x}=\hat{\mathbf{x}}} \quad \mathbf{J}_{kj} \leftarrow \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_j} \Big|_{\mathbf{x}=\hat{\mathbf{x}}}$ 
7      $\mathbf{H}_{ii} \leftarrow \mathbf{H}_{ii} + \mathbf{J}_{ki}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{ki} \quad \mathbf{H}_{ij} \leftarrow \mathbf{H}_{ij} + \mathbf{J}_{ki}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kj}$ 
8      $\mathbf{H}_{ji} \leftarrow \mathbf{H}_{ji} + \mathbf{J}_{kj}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{ki} \quad \mathbf{H}_{jj} \leftarrow \mathbf{H}_{jj} + \mathbf{J}_{kj}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kj}$ 
9      $\mathbf{b}_i \leftarrow \mathbf{b}_i + \mathbf{e}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{ki} \quad \mathbf{b}_j \leftarrow \mathbf{b}_j + \mathbf{e}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kj}$ 
10   $\Delta \mathbf{x}^{gn} \leftarrow \text{solve}(\mathbf{H}\Delta \mathbf{x} = -\mathbf{b})$ 
11   $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \Delta \mathbf{x}^{gn}$ 
12  $\mathbf{x}^* \leftarrow \hat{\mathbf{x}}$ 
13  $\mathbf{H}^* \leftarrow \mathbf{H}$ 
14 return  $\mathbf{H}^*, \mathbf{x}^*$ 

```

Gradient Descent: The Gradient Descent (GD) [39] method estimates the perturbation along the steepest direction of the gradient. It is very similar to GN, but it adds a parameter λ to control how far the current estimate should be moved along the direction of the perturbation.

$$\Delta \mathbf{x}^{gd} = -\lambda \frac{\partial L(\Delta \mathbf{x})_{\hat{\mathbf{x}}}}{\partial \Delta \mathbf{x}} \bigg|_{\Delta \mathbf{x}=0} = -\lambda [2\mathbf{b} + 2\mathbf{H}\Delta \mathbf{x}] \bigg|_{\Delta \mathbf{x}=0} = -\lambda \mathbf{b} \quad (2.26)$$

where $\mathbf{b} = \mathbf{e}^T \mathbf{\Omega}_z \mathbf{J}$, as shown in Eq. (2.22). At each step, λ is computed in the following manner:

$$\lambda^{gd} = \underset{\lambda > 0}{\operatorname{argmin}} [F(\hat{\mathbf{x}} + \Delta \mathbf{x}^{gd})] \quad (2.27)$$

The computed λ^{gd} needs to be calibrated to balance between safe updates and convergence speed. For this reason, GD suffers from slow convergence near the minimum.

Algorithm 2: Gradient Descent

```

1 Input:  $\hat{\mathbf{x}}, \mathcal{C} = \{(\mathbf{z}_k, \mathbf{\Omega}_{z_k})\}$  // Initial Guess and Constraints
2 while !converged do
3    $\mathbf{b} \leftarrow \mathbf{0}$ 
4   for all  $k \in \{0, \dots, t\}$  do
5      $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j) \leftarrow \mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k$ 
6      $\mathbf{J}_{ki} \leftarrow \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_i} \bigg|_{\mathbf{x}=\hat{\mathbf{x}}} \quad \mathbf{J}_{kj} \leftarrow \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_j} \bigg|_{\mathbf{x}=\hat{\mathbf{x}}}$ 
7      $\mathbf{b}_i \leftarrow \mathbf{b}_i + \mathbf{e}_{kij}^T \mathbf{\Omega}_{z_k} \mathbf{J}_{ki} \quad \mathbf{b}_j \leftarrow \mathbf{b}_j + \mathbf{e}_{kij}^T \mathbf{\Omega}_{z_k} \mathbf{J}_{kj}$ 
8    $\lambda^{gd*} = \underset{\lambda > 0}{\operatorname{argmin}} [F(\hat{\mathbf{x}} - \lambda^{gd*} \mathbf{b})]$ 
9    $\Delta \mathbf{x}^{gd} \leftarrow -\lambda^{gd*} \mathbf{b}$ 
10   $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \Delta \mathbf{x}^{gd}$ 
11  $\mathbf{x}^* \leftarrow \hat{\mathbf{x}}$ 
12 return  $\mathbf{x}^*$ 

```

Levenberg-Marquardt: The Levenberg-Marquardt (LM) [37] algorithm combines the advantages of both GN and GD. LM checks if the quadratic approximation $L(\Delta \mathbf{x})_{\hat{\mathbf{x}}}$ made by GN fits locally with the original cost function $F(\mathbf{x})$. If it does, then the method proceeds using GN; otherwise, the slower but more effective GD is used instead. To achieve this, Eq. (2.24) is modified in the following manner:

$$(\mathbf{H} + \alpha \mathbf{I}) \Delta \mathbf{x}^{lm} = -\mathbf{b} \quad (2.28)$$

For $\alpha = 0$, the method reverts to GN, while for large α , the contribution of $\mathbf{H}$ becomes negligible, resulting in $\Delta \mathbf{x}^{lm} = -\frac{1}{\alpha} \mathbf{b}$, where $\lambda = \frac{1}{\alpha}$, meaning the GD update is used instead.

The key difference between LM and GN is that the former rejects updates that would cause $F(\hat{\mathbf{x}} + \Delta\mathbf{x}) - F(\hat{\mathbf{x}}) > 0$. When this happens, it means that the quadratic approximation $L(\Delta\mathbf{x})_{\hat{\mathbf{x}}}$ does not represent the nonlinear cost function $F(\mathbf{x})$ well around the linearization point $\hat{\mathbf{x}}$, and smaller update steps are required. This is achieved by increasing the value of α , leading to more conservative behavior like GD. If $F(\hat{\mathbf{x}} + \Delta\mathbf{x}) - F(\hat{\mathbf{x}}) < 0$, then we want to speed up the convergence, and to do so, α is reduced, exhibiting a more optimistic behavior like GN. The different steps are repeated until convergence.

Algorithm 3: Levenberg-Marquardt

```

1 Input:  $\hat{\mathbf{x}}, \alpha_0, \mathcal{C} = \{(\mathbf{z}_k, \mathbf{\Omega}_{\mathbf{z}_k})\}$  // Initial Guess and Constraints
2  $v \leftarrow 2, \alpha \leftarrow \alpha_0$ 
3 while !converged do
4    $\mathbf{H}, \mathbf{b} \leftarrow \mathbf{0}$ 
5   for all  $k \in \{0, \dots, t\}$  do
6      $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j) \leftarrow \mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_k$ 
7      $\mathbf{J}_{ki} \leftarrow \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_i} \Big|_{\mathbf{x}=\hat{\mathbf{x}}} \quad \mathbf{J}_{kj} \leftarrow \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_j} \Big|_{\mathbf{x}=\hat{\mathbf{x}}}$ 
8      $\mathbf{H}_{ii} \leftarrow \mathbf{J}_{ki}^T \mathbf{\Omega}_{\mathbf{z}_k} \mathbf{J}_{ki} \quad \mathbf{H}_{ij} \leftarrow \mathbf{J}_{ki}^T \mathbf{\Omega}_{\mathbf{z}_k} \mathbf{J}_{kj}$ 
9      $\mathbf{H}_{ji} \leftarrow \mathbf{J}_{kj}^T \mathbf{\Omega}_{\mathbf{z}_k} \mathbf{J}_{ki} \quad \mathbf{H}_{jj} \leftarrow \mathbf{J}_{kj}^T \mathbf{\Omega}_{\mathbf{z}_k} \mathbf{J}_{kj}$ 
10     $\mathbf{b}_i \leftarrow \mathbf{b}_i + \mathbf{e}_{kij}^T \mathbf{\Omega}_{\mathbf{z}_k} \mathbf{J}_{ki} \quad \mathbf{b}_j \leftarrow \mathbf{b}_j + \mathbf{e}_{kij}^T \mathbf{\Omega}_{\mathbf{z}_k} \mathbf{J}_{kj}$ 
11   $\Delta\mathbf{x}^{lm} \leftarrow \text{solve}([\mathbf{H} + \alpha\mathbf{I}]\Delta\mathbf{x} = -\mathbf{b})$ 
12   $\rho = \frac{F(\hat{\mathbf{x}}) - F(\hat{\mathbf{x}} + \Delta\mathbf{x}^{lm})}{L(0)_{\hat{\mathbf{x}}} - L(\Delta\mathbf{x}^{lm})_{\hat{\mathbf{x}}}}$ 
13  if  $\rho > 0$  then
14     $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \Delta\mathbf{x}^{lm}$ 
15     $\alpha \leftarrow \alpha \cdot \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$ 
16     $v \leftarrow 2$ 
17  else
18     $\alpha \leftarrow \alpha v$ 
19     $v \leftarrow 2v$ 
20  $\mathbf{x}^* \leftarrow \hat{\mathbf{x}}$ 
21 return  $\mathbf{x}^*$ 

```

Powell's Dog Leg: The Powell's Dog Leg (PDL) [38] algorithm addresses the major disadvantage of LM, which involves recomputing the matrices $\mathbf{H}$ and $\mathbf{b}$ when an update step is rejected. Instead, PDL computes GN and GD updates accordingly and combines them appropriately, as even when a step is rejected, the direction of the step remains valid. Hence, each update step does not require recomputing the solution to $(\mathbf{H} + \alpha\mathbf{I})\Delta\mathbf{x}^{lm} = -\mathbf{b}$ multiple times for different values of α . PDL uses the concept of a *trust region*, which is defined as a sphere of radius r centered around the linearization point $\hat{\mathbf{x}}$. The trust region defines a neighborhood around $\hat{\mathbf{x}}$

in which $L(\Delta\mathbf{x})_{\hat{\mathbf{x}}}$ is a good approximation of $F(\mathbf{x})$. The interpolation between the GN and GD updates, respectively $\Delta\mathbf{x}^{gn}$ and $\Delta\mathbf{x}^{gd}$, is guided by the trust region. Similar to LM, if the GN updates fall within the trust region, they are preferred due to their superior speed; otherwise, GD updates are used instead. Based on the trust region defined by r , the PDL updates are computed in the following manner:

$$\Delta\mathbf{x}^{dl} = \begin{cases} \Delta\mathbf{x}^{gn}, & \text{if } \|\Delta\mathbf{x}^{gn}\| \leq r \\ \frac{r}{\|\Delta\mathbf{x}^{gd}\|} \Delta\mathbf{x}^{gd}, & \text{if } \|\Delta\mathbf{x}^{gd}\| \geq r \\ \Delta\mathbf{x}^{gd} + \beta(\Delta\mathbf{x}^{gn} - \Delta\mathbf{x}^{gd}), & \text{otherwise} \end{cases} \quad (2.29)$$

In the case where $\|\Delta\mathbf{x}^{gd}\| \geq r$, it is re-scaled in such a way that the update reaches the border of the trust region. In the other case, the two updates are linearly combined using the parameter β , which is computed so that $\|\Delta\mathbf{x}^{dl}\| = r$. During the different iterations of the algorithm, the radius r of the trust region changes. These changes are made based on how well $L(\Delta\mathbf{x})_{\hat{\mathbf{x}}}$ approximates $F(\mathbf{x})$ around the linearization point $\hat{\mathbf{x}}$, and the quality of this approximation is evaluated using the *gain ratio*:

$$\rho = \frac{F(\hat{\mathbf{x}}) - F(\hat{\mathbf{x}} + \Delta\mathbf{x}^{dl})}{L(0)_{\hat{\mathbf{x}}} - L(\Delta\mathbf{x}^{dl})_{\hat{\mathbf{x}}}} \quad (2.30)$$

which compares the reduction of the original function $F(\mathbf{x})$ and its quadratic approximation $L(\Delta\mathbf{x})$ when updating the current estimate using $\Delta\mathbf{x}^{dl}$. $\rho \rightarrow 1$ indicates that $L(\Delta\mathbf{x})_{\hat{\mathbf{x}}}$ is a good approximation of $F(\mathbf{x})$ around the linearization point $\hat{\mathbf{x}}$, which implies that the radius r of the trust region can be increased to allow larger update steps. On the other hand, $\rho \rightarrow 0$ indicates the opposite, leading to a decrease in the radius r .

2.3.1 UNCERTAINTY AND NON-LINEAR LEAST SQUARES

NLLS does not only provide a good estimate of the state variable $\mathbf{x}$, but it also gives us information about the solution's uncertainty. To retrieve such information, we need to understand how uncertainty and the different components of NLLS are related to each other. Our final goal is to estimate the joint probability $p(\mathbf{z}, \mathbf{x})$, which is achieved using the chain rule:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z} | \mathbf{x})p(\mathbf{x}) \quad (2.31)$$

As we have done for the NLLS derivation, we can instead parametrize the state variable in terms of the increment $\Delta\mathbf{x}$, which allows us to use the first-order Taylor expansion to approximate the measurement function in the following way:

$$\mathbf{h}(\hat{\mathbf{x}} + \Delta\mathbf{x}) \approx \mathbf{h}(\hat{\mathbf{x}}) + \mathbf{J}\Delta\mathbf{x} = \mathbf{h}(\Delta\mathbf{x}) \quad (2.32)$$

Algorithm 4: Powell's Dog Leg

```

1 Input:  $\hat{\mathbf{x}}, r_0, \mathcal{C} = \{(\mathbf{z}_k, \boldsymbol{\Omega}_{z_k})\}$  // Initial Guess and Constraints
2  $r \leftarrow r_0$ 
3 while !converged do
4    $\Delta \mathbf{x}^{gn} \leftarrow \text{ComputeGNStep}(\hat{\mathbf{x}}, \mathcal{C})$  // Compute single GN increment
5    $\Delta \mathbf{x}^{gd} \leftarrow \text{ComputeGDStep}(\hat{\mathbf{x}}, \mathcal{C})$  // Compute single GD increment
6   // Compute Dog Leg increment
7    $\Delta \mathbf{x}^{dl} \leftarrow 0$ 
8   if  $\|\Delta \mathbf{x}^{gn}\| < r$  then
9      $\Delta \mathbf{x}^{dl} \leftarrow \Delta \mathbf{x}^{gn}$ 
10  else if  $\|\Delta \mathbf{x}^{gd}\| > r$  then
11     $\Delta \mathbf{x}^{dl} \leftarrow r \frac{\Delta \mathbf{x}^{gd}}{\|\Delta \mathbf{x}^{gd}\|}$ 
12  else
13     $\Delta \mathbf{x}^{dl} \leftarrow \Delta \mathbf{x}^{gd} + \beta(\Delta \mathbf{x}^{gn} - \Delta \mathbf{x}^{gd})$  //  $\beta$  chosen to satisfy  $\|\Delta \mathbf{x}^{dl}\| = r$ 
14  // Accept/Reject Update
15   $\rho \leftarrow \frac{F(\hat{\mathbf{x}}) - F(\hat{\mathbf{x}} + \Delta \mathbf{x}^{dl})}{L(0)_{\hat{\mathbf{x}}} - L(\Delta \mathbf{x}^{dl})_{\hat{\mathbf{x}}}}$ 
16  if  $\rho > 0$  then
17     $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \Delta \mathbf{x}^{dl}$ 
18  // Compute radius Trust-Region
19  if  $\rho > 0.75$  then
20     $r \leftarrow \max(r, 3\|\Delta \mathbf{x}^{dl}\|)$ 
21  else if  $\rho < 0.25$  then
22     $r \leftarrow \frac{r}{2}$ 
23  $\mathbf{x}^* \leftarrow \hat{\mathbf{x}}$ 
24 return  $\mathbf{x}^*$ 

```

where $\mathbf{J}$ is the Jacobian, defined in Eq. (2.21). The change of variable simplifies the derivation because both $\mathbf{x}$ and $\Delta \mathbf{x}$ are normally distributed: $\mathbf{x} \approx \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Omega}_{\mathbf{x}}^{-1})$, and $\Delta \mathbf{x} \approx \mathcal{N}(0, \boldsymbol{\Omega}_{\Delta \mathbf{x}}^{-1})$. Here, the covariance matrices $\boldsymbol{\Omega}_{\mathbf{x}}^{-1}$ and $\boldsymbol{\Omega}_{\Delta \mathbf{x}}^{-1}$ are set to infinity since we lack prior information, causing the distributions to degenerate into uniform distributions. Consequently, both variables effectively share the same distribution, with the only distinction being that the Gaussian distribution of $\Delta \mathbf{x}$ is conveniently centered at zero, facilitating the derivation process. Passing from one parametrization to the other is achieved by summing or subtracting the mean $\boldsymbol{\mu}_{\mathbf{x}}$. So now the goal is to estimate the following joint distribution:

$$\begin{aligned}
 p(\hat{\mathbf{x}} + \Delta \mathbf{x}, \mathbf{z}) &= p(\mathbf{z} \mid \hat{\mathbf{x}} + \Delta \mathbf{x})p(\hat{\mathbf{x}} + \Delta \mathbf{x}) \\
 &= p(\mathbf{z} \mid \Delta \mathbf{x})p(\Delta \mathbf{x}) = p(\Delta \mathbf{x}, \mathbf{z})
 \end{aligned} \tag{2.33}$$

As $p(\mathbf{z} \mid \Delta\mathbf{x}) \approx \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}}, \boldsymbol{\Omega}_{\mathbf{z}|\mathbf{x}}^{-1})$, and $\mathbf{h}(\Delta\mathbf{x})$ is an affine transformation on $\Delta\mathbf{x} \approx \mathcal{N}(0, \boldsymbol{\Omega}_{\mathbf{x}}^{-1})$, we can fully describe the normally distributed $p(\mathbf{z} \mid \Delta\mathbf{x})$ in the following manner:

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} &= \mathbf{h}(\Delta\mathbf{x}) = \mathbf{h}(\hat{\mathbf{x}}) + \mathbf{J}\Delta\mathbf{x} \\ \boldsymbol{\Omega}_{\mathbf{z}|\mathbf{x}} &= \mathbf{J}\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{J}^T\end{aligned}\tag{2.34}$$

As each component of Eq. (2.33) has been defined, we can now derive the joint probability $p(\Delta\mathbf{x}, \mathbf{z})$, which is a Gaussian characterized by the following block structure:

$$\begin{aligned}p(\Delta\mathbf{x}, \mathbf{z}) &\approx \mathcal{N}(\boldsymbol{\mu}_{\Delta\mathbf{x}, \mathbf{z}}, \boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}}^{-1}) \\ \text{with } \boldsymbol{\mu}_{\Delta\mathbf{x}, \mathbf{z}} &= \begin{bmatrix} \mathbf{0} \\ \mathbf{h}(\hat{\mathbf{x}}) \end{bmatrix}, \quad \boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}} = \begin{bmatrix} \boldsymbol{\Omega}_{\Delta\mathbf{x}, \Delta\mathbf{x}} & \boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}} \\ \boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}}^T & \boldsymbol{\Omega}_{\mathbf{z}, \mathbf{z}} \end{bmatrix}\end{aligned}\tag{2.35}$$

where the values of the information matrix $\boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}}$ are obtained following [40]:

$$\begin{aligned}\boldsymbol{\Omega}_{\Delta\mathbf{x}, \Delta\mathbf{x}} &= \mathbf{J}\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{J}^T + \boldsymbol{\Omega}_{\Delta\mathbf{x}} = \mathbf{J}\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{J}^T \\ \boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}} &= -\mathbf{J}^T\boldsymbol{\Omega}_{\mathbf{z}} \\ \boldsymbol{\Omega}_{\mathbf{z}, \mathbf{z}} &= \boldsymbol{\Omega}_{\mathbf{z}}\end{aligned}\tag{2.36}$$

where $\mathbf{J}\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{J}^T + \boldsymbol{\Omega}_{\Delta\mathbf{x}} = \mathbf{J}\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{J}^T$ because $\Sigma_{\Delta\mathbf{x}} = \infty$, leading to $\Sigma_{\Delta\mathbf{x}}^{-1} = \boldsymbol{\Omega}_{\Delta\mathbf{x}} = 0$. We can further obtain the information vector $\mathbf{v}_{\Delta\mathbf{x}, \mathbf{z}}$ of Gaussian distribution $p(\Delta\mathbf{x}, \mathbf{z})$, as it is related to $\boldsymbol{\mu}_{\Delta\mathbf{x}, \mathbf{z}}$ by the following relation:

$$\mathbf{v}_{\Delta\mathbf{x}, \mathbf{z}} = \boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}}\boldsymbol{\mu}_{\Delta\mathbf{x}, \mathbf{z}} = \begin{bmatrix} \mathbf{v}_{\Delta\mathbf{x}} \\ \mathbf{v}_{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} -\mathbf{J}^T\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{h}(\hat{\mathbf{x}}) \\ \boldsymbol{\Omega}_{\mathbf{z}}\mathbf{h}(\hat{\mathbf{x}}) \end{bmatrix}\tag{2.37}$$

Now that the joint distribution has been obtained, we can condition on the observations $\mathbf{z}$ to finally link the posterior $p(\Delta\mathbf{x} \mid \mathbf{z})$ to the elements of NLLS. Following the results from [40], the conditioning is easily obtained using the information vectors:

$$\begin{aligned}\mathbf{v}_{\Delta\mathbf{x}|\mathbf{z}} &= \mathbf{v}_{\Delta\mathbf{x}} - \boldsymbol{\Omega}_{\Delta\mathbf{x}, \mathbf{z}}\mathbf{z} = -\mathbf{J}^T\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{h}(\hat{\mathbf{x}}) + \mathbf{J}^T\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{z} = -\mathbf{J}^T\boldsymbol{\Omega}_{\mathbf{z}}(\mathbf{h}(\hat{\mathbf{x}}) - \mathbf{z}) = -\mathbf{J}^T\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{e}(\hat{\mathbf{x}}) \\ \boldsymbol{\Omega}_{\Delta\mathbf{x}|\mathbf{z}} &= \boldsymbol{\Omega}_{\Delta\mathbf{x}, \Delta\mathbf{x}} = \mathbf{J}\boldsymbol{\Omega}_{\mathbf{z}}\mathbf{J}^T\end{aligned}\tag{2.38}$$

where

$$\begin{aligned}
\Omega_{\Delta\mathbf{x}|\mathbf{z}} &= \Omega_{\Delta\mathbf{x},\Delta\mathbf{x}} = \mathbf{J}\Omega_{\mathbf{z}}\mathbf{J}^T \\
&= (\mathbf{J}_1 \quad \mathbf{J}_2 \quad \dots \quad \mathbf{J}_k) \begin{pmatrix} \Omega_{\mathbf{z}_1} & & & \\ & \Omega_{\mathbf{z}_2} & & \\ & & \ddots & \\ & & & \Omega_{\mathbf{z}_k} \end{pmatrix} \begin{pmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_k \end{pmatrix} \\
&= \mathbf{J}_1\Omega_{\mathbf{z}_1}\mathbf{J}_1^T + \mathbf{J}_2\Omega_{\mathbf{z}_2}\mathbf{J}_2^T + \dots + \mathbf{J}_k\Omega_{\mathbf{z}_k}\mathbf{J}_k^T \\
&= \sum_{i=1}^k \mathbf{J}_i\Omega_{\mathbf{z}_i}\mathbf{J}_i^T \\
&= \sum_{i=1}^k \mathbf{H}_i \\
&= \mathbf{H}
\end{aligned} \tag{2.39}$$

where $\mathbf{H}$ is the one previously defined in Eq. (2.24). The mean $\boldsymbol{\mu}_{\Delta\mathbf{x}|\mathbf{z}}$ is obtained using the information vector $\mathbf{v}_{\Delta\mathbf{x}|\mathbf{z}}$ computed using Eq. (2.38):

$$\boldsymbol{\mu}_{\Delta\mathbf{x}|\mathbf{z}} = \Omega_{\Delta\mathbf{x}|\mathbf{z}}^{-1} \mathbf{v}_{\Delta\mathbf{x}|\mathbf{z}} = -\mathbf{H}^{-1} \mathbf{J}^T \Omega_{\mathbf{z}} \mathbf{e}(\hat{\mathbf{x}}) = -\mathbf{H}^{-1} \mathbf{b} \tag{2.40}$$

Eq. (2.40) demonstrates a fundamental result: the mean $\boldsymbol{\mu}_{\Delta\mathbf{x}|\mathbf{z}}$ is also the optimal solution to the NLLS, computed by iteratively solving Eq. (2.24). Furthermore, NLLS also provides a measure of uncertainty related to the solution, as $\Omega_{\Delta\mathbf{x}|\mathbf{z}} = \Omega_{\Delta\mathbf{x},\Delta\mathbf{x}} = \mathbf{H}$.

2.3.2 LEVERAGING SPARSITY

When using the PG formulation, important considerations can be made about the structure of the different matrices that need to be computed. From the derivation obtained in Eq. (2.22), we can observe that each constraint will contribute to the system as an element of the summation. As $\mathbf{b}_{kij} = \mathbf{e}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kij}$, and $\mathbf{H}_{kij} = \mathbf{J}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kij}$, their structure mainly depends on the corresponding Jacobian $\mathbf{J}_{kij}$. In the case of PG, the observation function $\mathbf{h}_k(\mathbf{x}_i, \mathbf{x}_j)$, and the corresponding error function $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)$ depend only on the state variables $\mathbf{x}_i$ and $\mathbf{x}_j$, so the Jacobian $\mathbf{J}_{kij}$ will have the following structure:

$$\mathbf{J}_{kij} = (\mathbf{0} \dots \mathbf{J}_{ki} \dots \mathbf{J}_{kj} \dots \mathbf{0}) \tag{2.41}$$

where the only two non-zero blocks of $\mathbf{J}_{kij}$ correspond to jacobians $\mathbf{J}_{ki} = \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_i} \Big|_{\mathbf{x}=\hat{\mathbf{x}}}$ and $\mathbf{J}_{kj} = \frac{\partial \mathbf{e}_{kij}(\mathbf{x})}{\partial \mathbf{x}_j} \Big|_{\mathbf{x}=\hat{\mathbf{x}}}$, respectively, of $\mathbf{x}_i$ and $\mathbf{x}_j$. From Eq. (2.41), the sparse structure of both

$\mathbf{b}_{kij}$ and $\mathbf{H}_{kij}$ are revealed:

$$\mathbf{b}_{kij}^T = (0 \cdots \mathbf{e}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{ki} \cdots \mathbf{e}_{kij}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kj} \cdots 0)^T$$

$$\mathbf{H}_{kij} = \begin{pmatrix} \ddots & & & & \\ & \mathbf{J}_{ki}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{ki} & \cdots & \mathbf{J}_{ki}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kj} & \\ & \vdots & \ddots & \vdots & \\ & \mathbf{J}_{kj}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{ki} & \cdots & \mathbf{J}_{kj}^T \Omega_{\mathbf{z}_k} \mathbf{J}_{kj} & \\ & & & \ddots & \end{pmatrix} \quad (2.42)$$

So each error $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)$ contributes four terms to the system matrix as $\mathbf{H} = \sum_{k=1}^t \mathbf{H}_{kij}$, which is then sparse by definition. Fig. 2.7 provides an effective visualization of the process of filling $\mathbf{H}$ and $\mathbf{b}$. The natural sparsity of $\mathbf{H}$ in the case of PG allows for efficiently solving

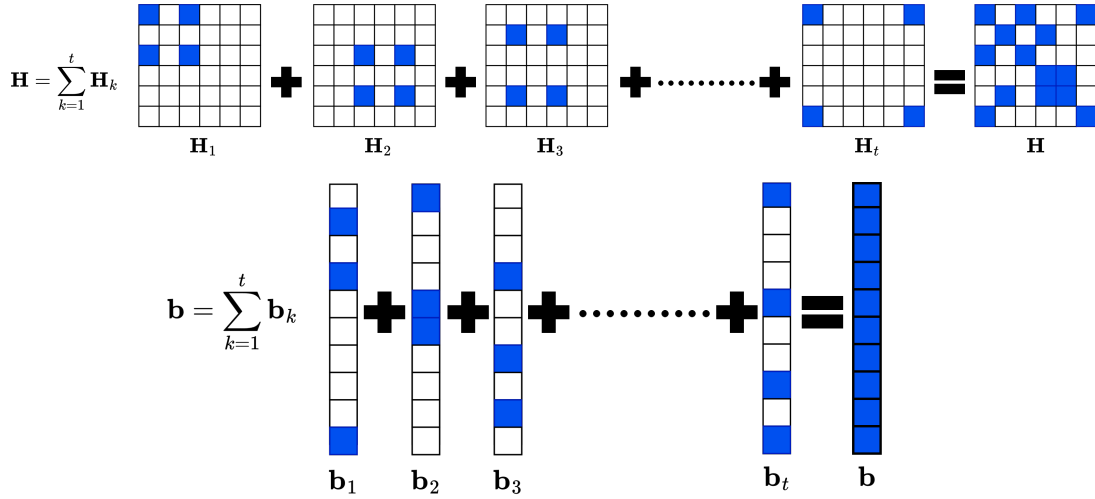


Figure 2.7: The top image illustrates how each constraint affects the $\mathbf{H}$ matrix, resulting in a sparse structure when dealing with PG. The bottom image shows how each constraint influences the vector $\mathbf{b}$, affecting only a few components at a time.

the linear system in Eq. (2.24) using iterative techniques, such as Preconditioned Conjugate Gradient (PCG) [41], or direct techniques, such as the Cholesky or QR decompositions. For details on these techniques, the reader can refer to the works in [42, 43]. As the errors depend only on the relative poses between $\mathbf{x}_i$ and $\mathbf{x}_j$, this leads to $F(\mathbf{x})$ being invariant to a rigid-body transformation on all the poses, which in other words means that the system $\mathbf{H}$ is underdetermined. The solution to this problem is to fix the value of one of the state variables, typically $\mathbf{x}_0$, to be zero. This can be practically executed in two ways: setting $\mathbf{H}_{0,0} = \mathbf{I}$ or suppressing the 0^{th} block-row and block-column.

2.3.3 RE-WEIGHTED NON-LINEAR LEAST SQUARES

NLLS [35] has the capability of handling the noise in the different measurements, as their weight is determined by their respective uncertainty. However, even if noisy, the measurements are assumed to always be generated by the same probability distribution, which in practice is often not true, especially when tackling SLAM (due to aliasing or wrong data association). The out-of-distribution measurements will be referred to as *outliers*, while the in-distribution ones will be referred to as *inliers*. As in NLLS, the errors are weighted quadratically, the gradients tend to minimize the measurements with high errors. High errors can be caused by: the initial guess being far away from the optimal one, or by having an outlier that is not correlated to the state variables. One simple solution consists of redefining the error terms in the following manner:

$$F(\mathbf{x}) = \sum_{k=1}^t \rho(\mathbf{u}_k(\mathbf{x}_i, \mathbf{x}_j)), \text{ with} \quad (2.43)$$

$$\mathbf{u}_k(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)^T \boldsymbol{\Omega}_{\mathbf{z}_k} \mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)}$$

where $\mathbf{u}_k(\mathbf{x}_i, \mathbf{x}_j)$ is the Mahalanobis distance [34], and $\rho(\cdot)$ is usually referred to as a robust cost function. Robust cost functions let us change the quadratic weighting of the error to reduce the influence of outliers. In the case of $\rho(\mathbf{u}_k(\mathbf{x}_i, \mathbf{x}_j)) = \mathbf{u}_k(\mathbf{x}_i, \mathbf{x}_j)^2$, we return to the classic NLLS formulation. To incorporate an arbitrary robust cost function in GN, LM, or PDL, we examine the effect of the robust cost functions on the original derivatives of the errors:

$$\begin{aligned} \frac{\partial \mathbf{u}(\mathbf{x})_{\boldsymbol{\Omega}_{\mathbf{z}}}}{\partial \mathbf{x}} &= \frac{\partial \|\mathbf{e}(\mathbf{x})\|_{\boldsymbol{\Omega}_{\mathbf{z}}}}{\partial \mathbf{x}} \\ &= \frac{1}{2\|\mathbf{e}(\mathbf{x})\|_{\boldsymbol{\Omega}_{\mathbf{z}}}} 2\mathbf{e}(\mathbf{x})^T \boldsymbol{\Omega}_{\mathbf{z}} \frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}} \\ &= \frac{1}{\mathbf{u}_k(\mathbf{x})} \frac{\partial \|\mathbf{e}(\mathbf{x})\|_{\boldsymbol{\Omega}_{\mathbf{z}}}^2}{\partial \mathbf{x}} \end{aligned} \quad (2.44)$$

We can then compute the derivative of a generic robust cost function using the chain rule on the following equation by leveraging Eq. (2.44), :

$$\begin{aligned} \frac{\partial \rho(u(\mathbf{x}))_{\boldsymbol{\Omega}_{\mathbf{z}}}}{\partial \mathbf{x}} &= \left. \frac{\partial \rho(u_{\boldsymbol{\Omega}_{\mathbf{z}}})}{\partial u} \right|_{u=u(\mathbf{x})} \frac{\partial u(\mathbf{x})_{\boldsymbol{\Omega}_{\mathbf{z}}}}{\partial \mathbf{x}} \\ &= \left. \frac{\partial \rho(u_{\boldsymbol{\Omega}_{\mathbf{z}}})}{\partial u} \right|_{u=u(\mathbf{x})} \frac{1}{\mathbf{u}_k(\mathbf{x})} \frac{\partial \|\mathbf{e}(\mathbf{x})\|_{\boldsymbol{\Omega}_{\mathbf{z}}}^2}{\partial \mathbf{x}} \\ &= \gamma(\mathbf{x}) \frac{\partial \|\mathbf{e}(\mathbf{x})\|_{\boldsymbol{\Omega}_{\mathbf{z}}}^2}{\partial \mathbf{x}} \end{aligned} \quad (2.45)$$

where $\gamma(\mathbf{x}) = \left. \frac{\partial \rho(u_{\boldsymbol{\Omega}_{\mathbf{z}}})}{\partial u} \right|_{u=u(\mathbf{x})} \frac{1}{\mathbf{u}_k(\mathbf{x})}$ is referred to as the robustifier, which is a scalar term that will reweight the information matrix. To pass from the NLLS to the Re-Weighted Non-

Linear Least Squares (RW-NLLS) [44] formulation, it is only necessary to substitute Ω_z with $\hat{\Omega}_z = \gamma(\mathbf{x})\Omega_z$. In the following table, the most commonly used robust cost functions and their corresponding robustifiers are reported:

When robustifiers are used, we lose all the relationships between uncertainty and NLLS, which

Type	$\rho(u(x))$	$\gamma(x)$
L_2	$\frac{u(x)^2}{2}$	1
L_1	$ u(x) $	$\frac{1}{u(x)}$
Fair	$c^2 \left[\frac{ u(x) }{c} - \log\left(1 + \frac{ u(x) }{c}\right) \right]$	$\frac{c}{c+ u(x) }$
Huber	$\begin{cases} \frac{u(x)^2}{2}, & u(x) \leq c \\ c(u(x) - \frac{c}{2}), & u(x) > c \end{cases}$	$\begin{cases} 1, & u(x) \leq c \\ \frac{c}{ u(x) }, & u(x) > c \end{cases}$
Cauchy	$\frac{c^2}{2} \log\left(1 + \frac{u(x)^2}{c^2}\right)$	$\frac{c^2}{c^2 + u(x)^2}$
Geman-McClure	$\frac{1}{2} \frac{u(x)^2}{1 + u(x)^2}$	$\frac{1}{(1 + u(x)^2)^2}$
Welsch	$\frac{c^2}{2} \left[1 - \exp\left(-\frac{u(x)^2}{c^2}\right) \right]$	$\exp\left(-\frac{u(x)^2}{c^2}\right)$
Tukey	$\begin{cases} \frac{c^2}{6} \left[1 - \left(1 - \left(\frac{u(x)^2}{c^2}\right)^3\right) \right], & u(x) \leq c \\ \frac{c^2}{6}, & u(x) > c \end{cases}$	$\begin{cases} \left[1 - \frac{u(x)^2}{c^2}\right]^2, & u(x) \leq c \\ 0, & u(x) > c \end{cases}$

Table 2.1: Table listing different types of robust cost functions $\rho(u(x))$ and their corresponding reweighting factors $\gamma(x)$.

were derived in the previous section. To recover those properties, first RW-NLLS is run to obtain a good solution and identify the outliers, and then NLLS is run only on the inlier measurements.

2.3.4 OPTIMIZATION ON MANIFOLDS

The different optimization algorithms presented in Sec. (2.3) assume that we are operating in Euclidean spaces, as vector addition and subtraction, which are the operators that we use to apply the perturbation to the state variable $\mathbf{x}$, are defined only in those spaces. In SLAM, or more generally when solving robotics-related problems, it is common to encounter state variables that use overparameterized non-Euclidean representations instead of minimal Euclidean representations. The advantage of using such representations is that they avoid singularities which minimal representations are often subject to. This benefit comes at the cost of not being able to apply the updates in a straightforward way and adding degrees of freedom that could introduce errors in the solution, as opposed to the minimal representation. To obtain the benefits of both representations, it is necessary to perform the optimization on a manifold [45], which is a mathematical space that admits a local Euclidean parametrization around its elements. To apply the Euclidean perturbations $\Delta\mathbf{x} \in \mathbb{R}^d$ on a manifold point $\mathbf{X} \in \mathbb{M}$, it is necessary to de-

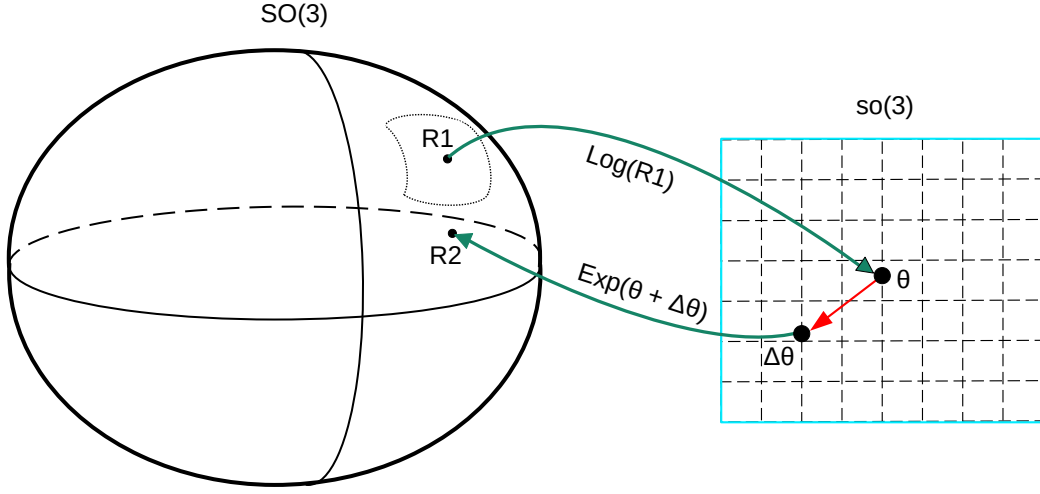


Figure 2.8: Image illustrating the manifold operations defined for $SO(3)$, enabling the application of an increment $\Delta\theta$, defined in the local Euclidean parameterization $\mathfrak{so}(3)$, to $R_1 \in SO(3)$.

fine operators that encapsulate the *charts* around $\mathbf{X}$. A chart is a function that maps the points living in $\mathbf{X}$'s neighborhood to the local Euclidean space centered around $\mathbf{X}$, and the inverse chart maps the points in the local Euclidean space around $\mathbf{X}$ back to the manifold, as shown in Fig. 2.8. The $\text{chart}_{\mathbf{X}}(\mathbf{X}) = \mathbf{0}$, as the mapping is centered on $\mathbf{X}$.

$$\begin{aligned} \text{chart}_{\mathbf{X}}(\cdot) : \mathbb{M} &\rightarrow \mathbb{R}^d \\ \text{chart}_{\mathbf{X}}^{-1}(\cdot) : \mathbb{R}^d &\rightarrow \mathbb{M} \end{aligned} \quad (2.46)$$

Here we define the two new operators that map a variation in the Euclidean space to a variation in the manifold space by using charts:

$$\begin{aligned} \boxplus : \mathbb{M} \times \mathbb{R}^d &\rightarrow \mathbb{M} \\ \boxminus : \mathbb{M} \times \mathbb{M} &\rightarrow \mathbb{R}^d \end{aligned} \quad (2.47)$$

The boxplus $\boxplus$ and the boxminus $\boxminus$ operators role is analogous to the sum and difference in the Euclidean space:

$$\begin{aligned} \mathbf{X}' &= \mathbf{X} \boxplus \Delta\mathbf{x} = \text{chart}_{\mathbf{X}}^{-1}(\Delta\mathbf{x}) \\ \Delta\mathbf{x} &= \mathbf{X}' \boxminus \mathbf{X} = \text{chart}_{\mathbf{X}}(\mathbf{X}') \end{aligned} \quad (2.48)$$

The boxplus operator $\boxplus$ encapsulates three different operations: computing $\text{chart}_{\mathbf{X}}(\cdot)$ around $\mathbf{X}$, applying the perturbation $\Delta\mathbf{x} \in \mathbb{R}^d$, and then using $\text{chart}_{\mathbf{X}}^{-1}(\cdot)$ to map it back to the corresponding new point on the manifold $\mathbf{X}'$. The boxminus operator $\boxminus$ also encapsulates three

different operations: computing $\text{chart}_{\mathbf{X}}(\cdot)$ around $\mathbf{X}$, mapping $\mathbf{X}'$ to the local Euclidean space around $\mathbf{X}$, and computing the difference $\Delta\mathbf{x}$ between the two. If $\mathbf{X}' = \mathbf{X}$, then that implies that $\Delta\mathbf{x} = 0$. These operators allow us to use the optimization algorithms even when using overparameterized representations of the state variables with minimal changes, as long as they lie on a smooth manifold. Here is how the different elements computed are modified with the introduction of manifolds and the different operators:

$$\begin{aligned} \mathbf{e}_k(\hat{\mathbf{X}} \boxplus \Delta\mathbf{x}) &= \mathbf{h}_k(\hat{\mathbf{X}} \boxplus \Delta\mathbf{x}) \boxminus \mathbf{Z}_k \\ \mathbf{e}_k(\hat{\mathbf{X}} \boxplus \Delta\mathbf{x}) &\approx \mathbf{e}_k(\hat{\mathbf{X}}) + \tilde{\mathbf{J}}_k \Delta\mathbf{x} \end{aligned} \quad (2.49)$$

where $\tilde{\mathbf{J}}_k$ will substitute the previously used $\mathbf{J}_k$, and is defined as follows:

$$\tilde{\mathbf{J}}_k = \left. \frac{\partial \mathbf{e}_k(\hat{\mathbf{X}} \boxplus \Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \right|_{\Delta\mathbf{x}=0} \quad (2.50)$$

A practical example of a non-Euclidean space is the Special Orthogonal Group in three dimensions or $\text{SO}(3)$ [46, 47], which is the space of 3D rotations: $\text{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3 \times 3} \mid \mathbf{R}^T \mathbf{R} = \mathbf{I} \wedge \det(\mathbf{R}) = 1\}$. An element $\mathbf{R} \in \text{SO}(3)$ is called a rotation matrix. The corresponding local Euclidean space $\text{so}(3)$ is referred to as the *Lie Algebra*, which is the space of three-dimensional skew-symmetric matrices that can be fully determined by the vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$, as the following relation exists:

$$[\boldsymbol{\theta}]_{\times} = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix} \quad (2.51)$$

The *exponential map* $\exp(\cdot) : \text{so}(3) \rightarrow \text{SO}(3)$ maps elements in the Lie Algebra to their corresponding rotations matrix using the Rodrigues formula [48]:

$$\exp(\boldsymbol{\theta}) = \mathbf{I} + \frac{\sin(\|\boldsymbol{\theta}\|)}{\|\boldsymbol{\theta}\|} [\boldsymbol{\theta}]_{\times} + \frac{1 - \cos(\|\boldsymbol{\theta}\|)}{\|\boldsymbol{\theta}\|^2} [\boldsymbol{\theta}]_{\times}^2 \quad (2.52)$$

While the inverse function of the exponential map will be referred to as the *logarithm map* $\log(\cdot) : \text{SO}(3) \rightarrow \text{so}(3)$, which maps rotation matrices to their corresponding skew-symmetric matrices that can be represented by the vector $\boldsymbol{\theta}$:

$$\begin{aligned} \log(\mathbf{R}) &= \frac{\phi(\mathbf{R} - \mathbf{R}^T)}{2 \sin(\phi)} \\ \text{where } \phi &= \cos^{-1} \frac{(\text{tr}(\mathbf{R}) - 1)}{2} \end{aligned} \quad (2.53)$$

with $\text{tr}(\mathbf{R})$ being the trace of the matrix $\mathbf{R}$, and $[\boldsymbol{\theta}]_{\times} = \log(\mathbf{R})$. Now that the direct and inverse mappings have been defined, we can define the corresponding $\boxplus$ and $\boxminus$ when handling rotation matrices, which allows us to use GN, GD, and PDL even in $\text{SO}(3)$:

$$\begin{aligned} \mathbf{R}_2 &= \mathbf{R}_1 \boxplus \Delta\boldsymbol{\theta} = \exp(\Delta\boldsymbol{\theta}) \mathbf{R}_1 \\ [\Delta\boldsymbol{\theta}]_{\times} &= \mathbf{R}_1 \boxminus \mathbf{R}_2 = \log(\mathbf{R}_1^T \mathbf{R}_2) \end{aligned} \quad (2.54)$$

2.4 FACTOR GRAPHS FOR ROBOTICS

Pose Graph are an instance of a more general graphical model named Factor Graph (FG). A PG is an FG in which the state variables are only poses, and the number of variables connected to a *factor*, which corresponds to our error model, is always two. FG are able to describe and represent an even wider category of problems compared to the PG, but at the cost of losing some useful properties of Pose Graph, such as the sparsity that allows for more efficient operations. For the FG, Eq. (2.17) becomes:

$$\mathbf{x}_{1:t}^* = \underset{\mathbf{x}_{1:t}}{\operatorname{argmax}} \prod_{k=1}^t p(\mathbf{z}_k \mid \{\mathbf{x}_i, \dots, \mathbf{x}_n\}) \quad (2.55)$$

where $\{\mathbf{x}_i, \dots, \mathbf{x}_n\}$ is the set of variables from which the observations $\mathbf{z}_k$ depend. Different problems found in robotics, and how to use FG and PG to model those problems, will be presented in the following paragraphs.

SLAM: In the Simultaneous Localization and Mapping problem, the robot needs to localize in a map that it is incrementally building while exploring the environment. Let us assume that for the map representation, a set of landmarks $\{\mathbf{l}_0, \dots, \mathbf{l}_{M-1}\}$ is used, where each landmark $\mathbf{l}_m \in \mathbb{R}^3$ (or $\mathbb{R}^2$). Then, solving the Simultaneous Localization and Mapping problem translates to jointly estimating the robot's poses $\{\mathbf{x}_1, \dots, \mathbf{x}_t\}$ and the landmarks' locations $\{\mathbf{l}_0, \dots, \mathbf{l}_{M-1}\}$. It can be represented by a graph, as shown in Fig. 2.9, by:

- binary errors between consecutive robot poses $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_{i+1})$ which correspond to the motion model $p(\mathbf{z}_k \mid \mathbf{x}_i, \mathbf{x}_{i+1})$;
- binary errors between non-consecutive robot poses $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_j)$ which correspond to place recognition system, which is modeled using $p(\mathbf{z}_k \mid \mathbf{x}_i, \mathbf{x}_j)$;
- binary errors between a robot pose and a landmark $\mathbf{e}_k(\mathbf{x}_i, \mathbf{l}_m)$ which correspond to a landmark detecting system, modeled by $p(\mathbf{z}_k \mid \mathbf{x}_i, \mathbf{l}_m)$;

Tracking: Here, the goal is tracking a target, which can be the same robot or even an external element of interest, such as humans the robot wants to interact with. Tracking translates to estimating the target's trajectory. The tracking problem can be represented by a FG, as shown in Fig. 2.10, with the following factors:

- binary errors between the consecutive target's poses $\mathbf{e}_k(\mathbf{x}_i, \mathbf{x}_{i+1})$ which correspond to the target's motion model $p(\mathbf{z}_k \mid \mathbf{x}_i, \mathbf{x}_{i+1})$;
- unary errors $\mathbf{e}_k(\mathbf{x}_i)$ which correspond to an observation of some known quality of the environment which gives information regarding the target's pose $\mathbf{x}_i$;

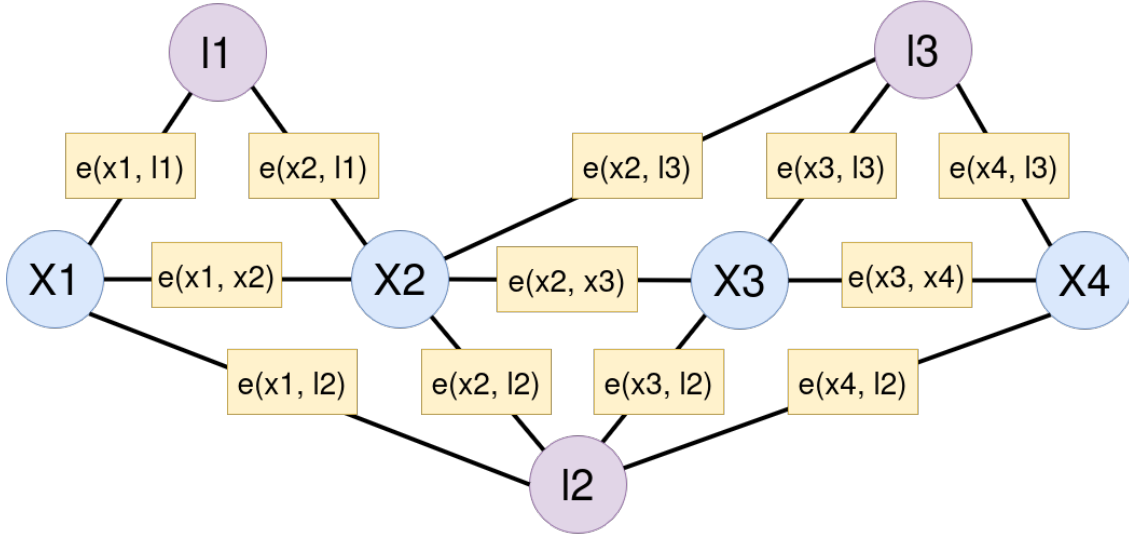


Figure 2.9: FG formulation for SLAM. The variables $\mathbf{x}_i$, corresponding to the robot state, are highlighted in blue, while the variables $\mathbf{l}_i$, corresponding to landmark states, are highlighted in purple.

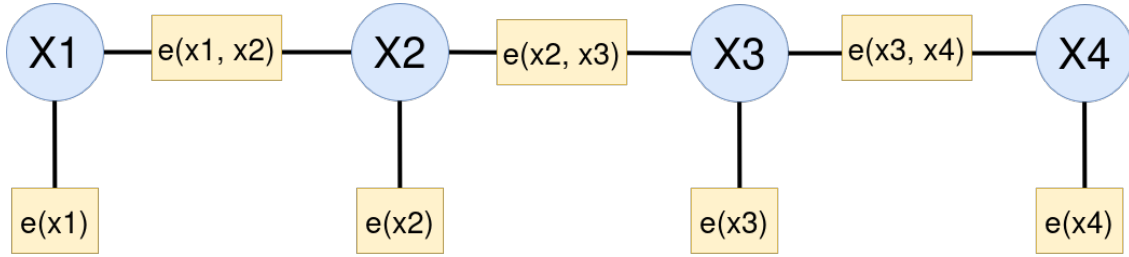


Figure 2.10: FG formulation for tracking.

Structure-from-Motion: Structure-from-Motion is the problem estimating the relative camera poses and the 3D structure of the environment from a set of images. Similar to SLAM, but there is no direct relation between cameras other than the common observed elements in the different images. Solving SfM translates to jointly estimating the poses of the cameras in the environment, and a set of points that encode the geometry of the observed environment. SfM can be represented by a FG, as shown in Fig. 2.11, with the following factors:

- binary errors between a camera and a landmark (point) $e_k(\mathbf{x}_i, \mathbf{l}_m)$ which represents the geometric relation between the camera pose and the observed point modeled by $p(\mathbf{z}_k | \mathbf{x}_i, \mathbf{l}_m)$;

Optimal Control: Optimal control is the process of determining a sequence of control inputs that minimizes a cost function, which describes a specific characteristic of the system. Here, we illustrate how a factor graph can be used to model one instance of optimal control—namely,

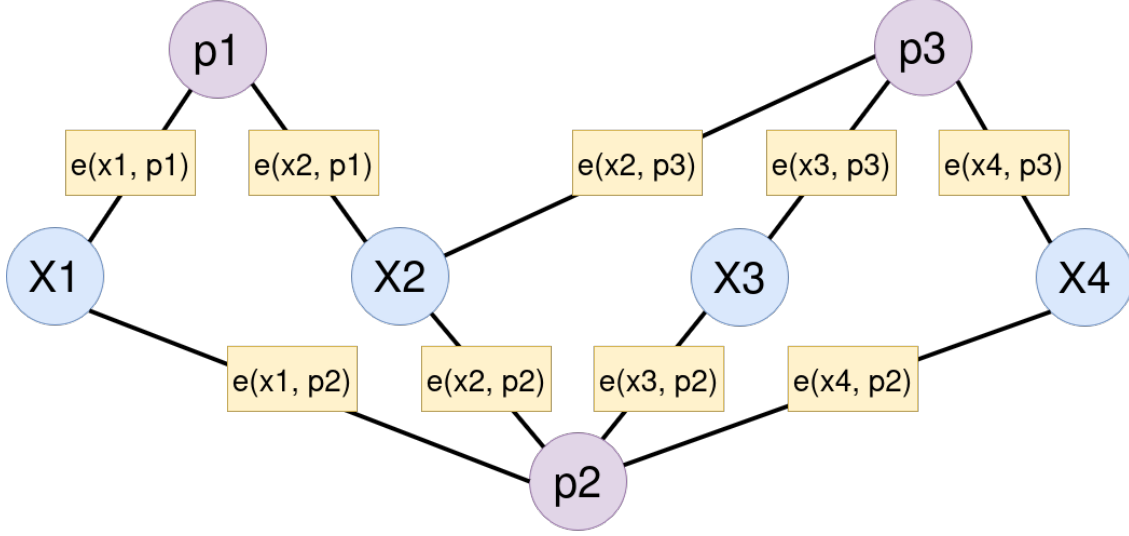


Figure 2.11: FG formulation for SfM. The variables $\mathbf{x}_i$, corresponding to the robot state, are highlighted in blue, while the variables $\mathbf{p}_i$, corresponding to points in the environment, are highlighted in purple.

Linear Quadratic Regulator (LQR). LQR derives the optimal gains for a linear system by minimizing a quadratic cost function that penalizes both control effort and state error. Solving LQR involves jointly estimating the desired states $\mathbf{x}_{1:t}$ and the corresponding control inputs $\mathbf{u}_{1:t}$ required to reach them. LQR can be represented by a FG, as shown in Fig. 2.12, which includes the following factors:

- Unary errors on the control inputs $\mathbf{u}_i$, which define the quadratic cost $e_k(\mathbf{u}_i)$ associated with control effort.
- Unary errors on the poses $\mathbf{x}_i$, which define the quadratic cost $e_k(\mathbf{x}_i)$ related to pose accuracy.
- Ternary errors between subsequent poses and the control input that relates them, $e_k(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1})$, representing the constraints imposed by the system dynamics.

Switching Systems: Switching systems are used to select the most appropriate model for the task or situation at hand. Here, we illustrate how a factor graph can be used to model one instance of a switching system—namely, Switching Linear Dynamic System (SLDS). SLDS is commonly used to approximate a complex nonlinear dynamic system with a set of linear systems. Solving SLDS involves jointly estimating the desired states $\mathbf{x}_{1:t}$ and selecting the most suitable linear system given the observations $\mathbf{z}_{1:t}$. SLDS can be represented by a FG, as shown in Fig. 2.13, which includes the following factors:

- Binary errors on the linear dynamics models $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$, which define the quadratic cost $e_k(\mathbf{q}_i, \mathbf{q}_{i+1})$.

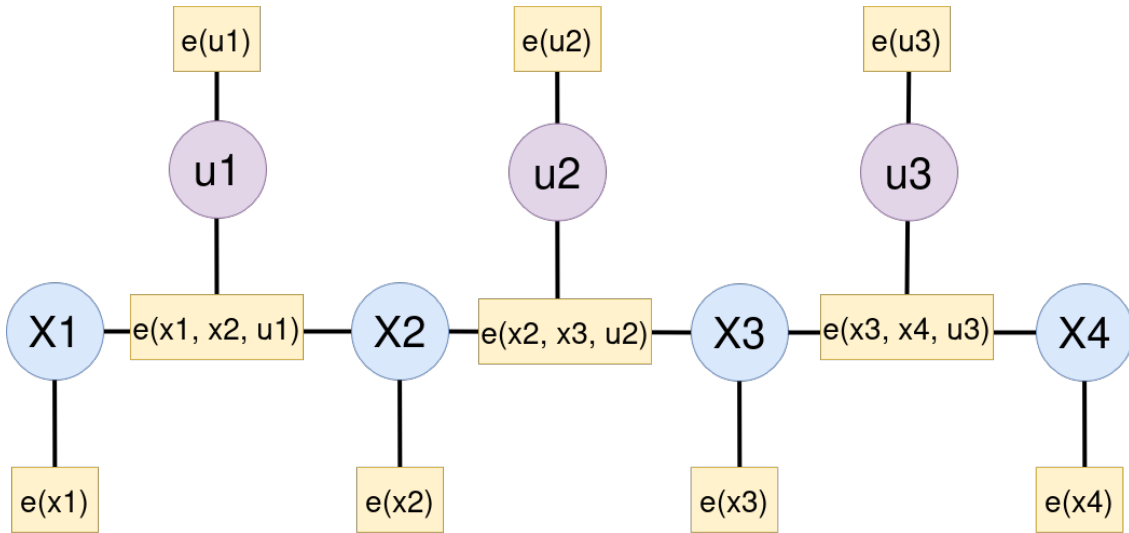


Figure 2.12: FG formulation for optimal control. The variables $\mathbf{x}_i$, corresponding to the robot state, are highlighted in blue, while the variables $\mathbf{u}_i$, corresponding to the optimal control inputs, are highlighted in purple.

- Unary errors $e_k(\mathbf{x}_i)$, which correspond to an observation of known quality from the environment that provides information regarding the target's pose $\mathbf{x}_i$.
- Ternary errors between subsequent poses and the linear system that relates them, $e_k(\mathbf{x}_i, \mathbf{q}_i, \mathbf{x}_{i+1})$, representing how well the chosen linear system approximates the actual dynamics.

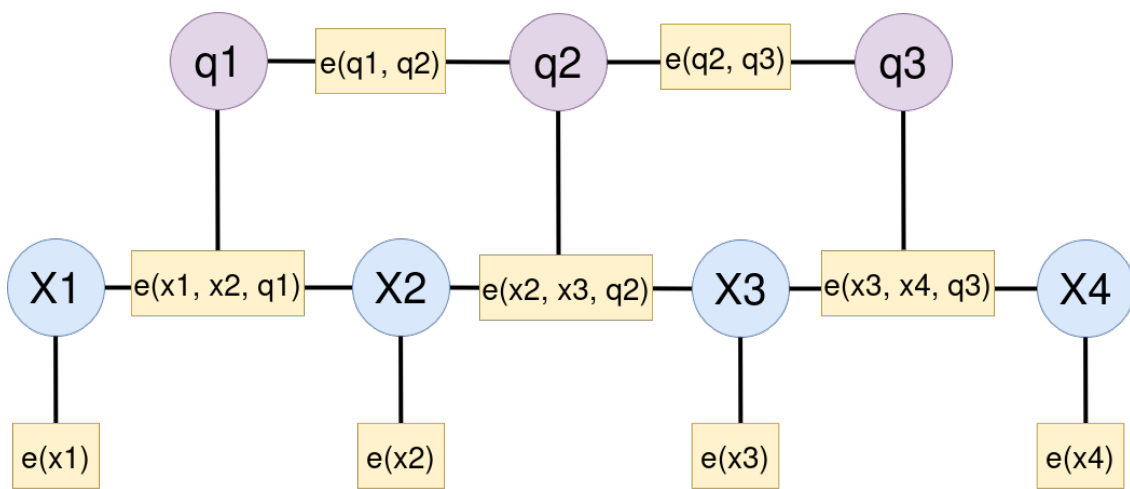


Figure 2.13: FG formulation for the switching systems problem. The variables $\mathbf{x}_i$, corresponding to the robot state, are highlighted in blue, while the variables $\mathbf{q}_i$, corresponding to the optimal linear dynamic system, are highlighted in purple.

3

A Graph-based Optimization Framework for Hand-Eye Calibration for Multi-Camera Setups

Pose Graphs (PGs) provide a versatile and powerful graphical framework for solving robotics problems that can be formulated as Maximum a Posteriori (MAP) estimation tasks. PGs ensure both accuracy and robustness as they natively incorporate the measurement’s noise characteristics in the optimization process. In this chapter, we showcase the expressiveness of PGs by applying them to the Hand-Eye (HE) calibration problem. HE involves estimating the spatial transformation between a reference frame—typically the base of a robotic arm—and the reference frame of one or multiple cameras, a configuration commonly found in industrial applications and mobile robots. This calibration task can be interpreted as a specific instance of the SLAM problem. This chapter builds on this insight and proposes a PG-based approach to HE calibration. We demonstrate the validity of this approach through comprehensive simulations and real-world experiments.

3.1 MOTIVATION

The ability to robustly perceive the surrounding environment is a key factor for the development and deployment of intelligent robotic work cells in industry. Perception devices, for example cameras, are widely used in industrial robotics applications to perform complex tasks that involves robot arms such as visual inspections [49, 50] and bin-picking [51, 52]. They are also involved as assistive devices that continuously monitor the position of human workers in collaborative tasks with robots [53, 54]. However, the information provided by the cameras is generally expressed with respect to their own coordinate systems. To make it possible to con-

to control a robot arm based on such visual information, it is first necessary to accurately calculate the pose of the camera relative to the robot gripper or relative to the robot base or work cell. This process is called *hand-eye calibration*, and this is defined as the procedure to determine the spatial transformation between a robot reference frame (e.g., robot base or end-effector frames) and the reference frame of the camera. The hand-eye calibration problem can be classified into

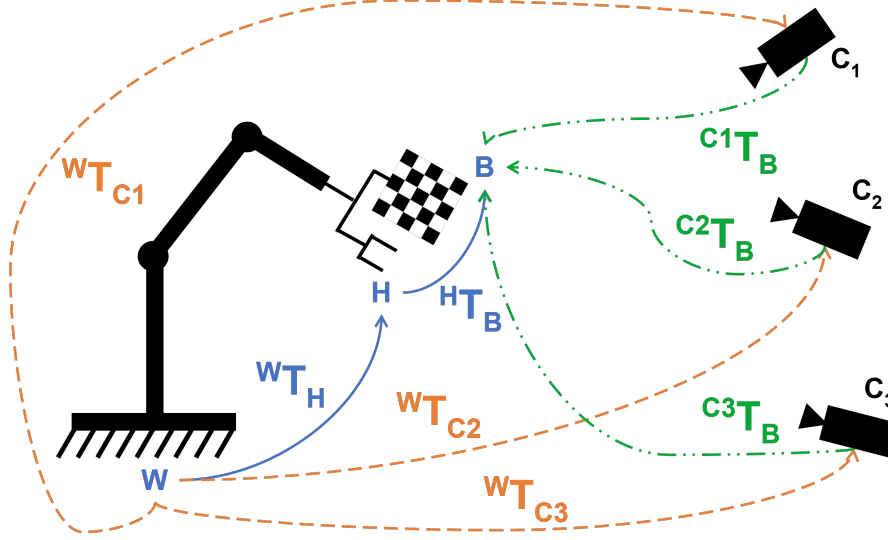


Figure 3.1: Multi-camera Eye-on-Base calibration setup. The goal of the proposed hand-eye calibration is to compute the position of each camera in the reference frame W , i.e., the transformations ${}^W T_{C1...C3}$ in the figure.

two categories[55]:

- **Eye-in-hand:** The camera is rigidly mounted on the robot's wrist;
- **Eye-on-base (or Eye-to-hand):** One or more cameras are fixed while the robot arm is moving (e.g., Fig. 3.1)

There exists a variety of methods that solve this problem, many of them based on a global optimization that minimizes specific quantities, such as geometric ones like translation and rotation errors or visual ones like reprojection errors. What, instead, is rarely done is to exploit the underlying graph structure of the problem. In fact, as introduced in [56], the HE calibration problem with an eye-in-hand setup can be seen as an instance of the SLAM problem and can be solved by exploiting standard probabilistic frameworks, such as pose graph optimization [33]. In this chapter, we exploit this intuition by proposing a general graph-based tool to solve the complementary eye-on-base calibration problem. To cast this problem to SLAM, one can consider that the motion is relative: instead of considering the motion of the end-effector in front of the static camera, one can imagine the camera is moving, and the robot's end-effector is static.

Thus, taking images (i.e., observations) while the robot moves a checkerboard mounted on the end-effector to different poses (i.e., landmarks) results in observations of landmarks used for the optimization. Our solution is based on the minimization of reprojection error, which is a well-known technique that has been applied to different types of problems (e.g. camera calibration, bundle adjustment, ...) but surprisingly rarely in the context of HE calibration problems. This technique also allows us to avoid relying strictly on the Perspective-n-Point (PnP) algorithm, which in our case is simply used to infer an initial guess for the optimization. The main contributions of this chapter* are the following:

- (i) The proposed approach extends a recent state-of-the-art approach [56] by implementing a graph-based optimization solution to *eye-on-base* setups with one camera (Fig. 3.2), not covered in the basic method;
- (ii) Our method also covers multi-camera setups (Fig. 3.1); thus, a general and unified graph-based optimization framework for hand-eye calibration is achieved;
- (iii) An exhaustive evaluation of the proposed method is reported. The evaluation is primarily performed in simulated environments, where the robustness to visual and geometric noise can be tested. The proposed approach outperforms several popular calibration approaches, proving to be robust to high levels of noise in the calibration data. We also tested our method on a real setup, confirming the convincing results obtained in the simulations;
- (iv) An open-source implementation of the proposed hand-eye calibration method is made publicly available at:
https://bitbucket.org/freelist/gm_handeye

3.2 RELATED WORK

In the literature, several methods have been proposed for HE calibration. Shiu and Ahmad [57] initially proposed a method for the estimation of the HE transformation based on the solution of a homogeneous equation. Tsai and Lenz [58] proposed a widely adopted calibration method (e.g., implemented in [59]) that first estimates the translation and then the rotation of a HE transformation. Similarly to Shiu and Tsai, Chou and Kamel [60] solved an estimation problem based on a normalized quaternion representation to transform the kinematic equation into two simple and structured linear systems with rank-deficient coefficient matrices. Daniilidis and Bayro-Corrochano [61] proposed a similar framework that is based on dual quaternion parameterization and singular value decomposition, and Srobl and Hirzinger [62]

*The results of this chapter are derived from: D. Evangelista, E. Olivastri, D. Allegro, E. Menegatti and A. Pretto, "A Graph-Based Optimization Framework for Hand-Eye Calibration for Multi-Camera Setups". 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom.

introduced a novel metric to optimize the estimation problem by appropriately weighting translation and rotation errors in the optimization problem. Park and Martin [63] estimated the HE transformation in the Euclidean group using Lie group and least-square optimization. Gwak *et al.* [64] proposed a cyclic coordinate descent algorithm to optimize objective functions in $SE(3)$ and applied it to a class of robotic problems such as HE calibration. In this direction, Horaud and Dornaika [65] proposed a linear formulation of the same optimization problem. The same method has been further applied to online calibration by Andreff *et al.* [66].

More recently, Shah [67] formulated a closed-form solution for the HE problem by using an SVD-based algorithm and the Kronecker product to solve for rotation and translation separately, while LI *et al.* [68] instead used dual quaternions to solve them simultaneously to overcome the limitations of the Kronecker product.

More general formulations of the HE calibration problem have been done in [69] and [70]. The latter, in particular, mathematically formulates the problem of camera-to-camera calibration using the geometric constraints of HE calibration when a common reference frame (e.g., robot base frame) is taken into the loop. Although this method formally extends the formulation of [58] to multi-camera setups, it requires an external motion capture system to accurately recover the position of cameras during calibration, making the whole approach not easily scalable or applicable to small setups.

In our previous work [56], we proposed a more general approach based on pose graph optimization that achieves high levels of accuracy and introduces a general solution that can be easily applied to different robotic setups and camera projection models (e.g., both standard pinhole camera projection and X-ray source detector models [71]). Although the proposed method is quite general, it does not cover eye-on-base setups in single-camera setups, and it does not handle multi-camera robotic systems. In this work, we overcome this limitation by extending the approach to eye-on-base setups; moreover, we give a general formulation applicable to multi-camera setups where the position of multiple cameras has to be computed w.r.t. a robot base frame.

3.3 METHOD

This section presents a detailed overview of the proposed approach. This framework extends [56] by providing HE calibration solutions for eye-on-base for single- and multi-camera setups. Together with [56], all the calibration needs that arise in robotic industrial applications can be fulfilled.

3.3.1 EYE-ON-BASE CALIBRATION FOR SINGLE CAMERA

Here, we report the necessary notation for eye-on-base single camera setups.

- $\mathbf{W}$: world reference frame, usually placed in the robot’s base;

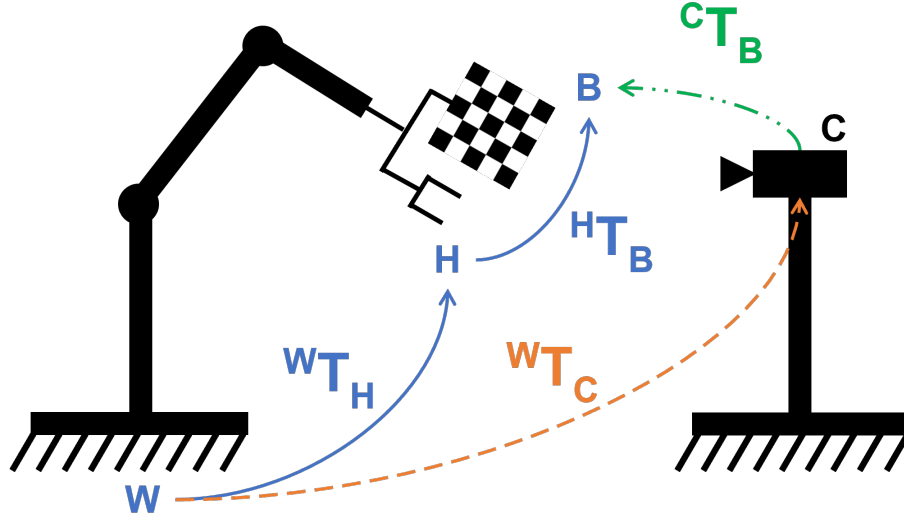


Figure 3.2: Single-camera Eye-on-Base calibration setup. The goal of HE calibration here is to compute the position of the camera in the W reference frame, namely the transformation ${}^W T_C$ in the figure.

- H : hand reference frame, placed in the robot's end-effector;
- B : checkerboard reference frame;
- C : camera reference frame;

While the transformations are :

- ${}^W T_H$: Isometry representing the pose of the end-effector in the world reference frame;
- ${}^H T_B$: Isometry representing the pose of the checkerboard in the end-effector reference frame;
- ${}^C T_B$: Isometry representing the pose of the checkerboard in the camera reference frame;
- ${}^W T_C$: Isometry representing the pose of the camera in the world reference frame.

Fig. 3.3 shows the graph structure for the eye-on-base calibration. Using the notation from [56]: circles represent the state variables that need to be estimated, while hexagons are fixed parameters such as K and $P_{1..Z}$, that represent the camera matrix and the set of corners points of the checkerboard in their reference system, respectively. M represents the number of observations, i.e., images and robot poses, used for the calibration. The rectangles represent the different error terms. The error is generally defined as

$$\mathbf{e}_i(\mathbf{x}) = \mathbf{h}_i(\mathbf{x}) \boxminus \mathbf{z}_i, \quad (3.1)$$

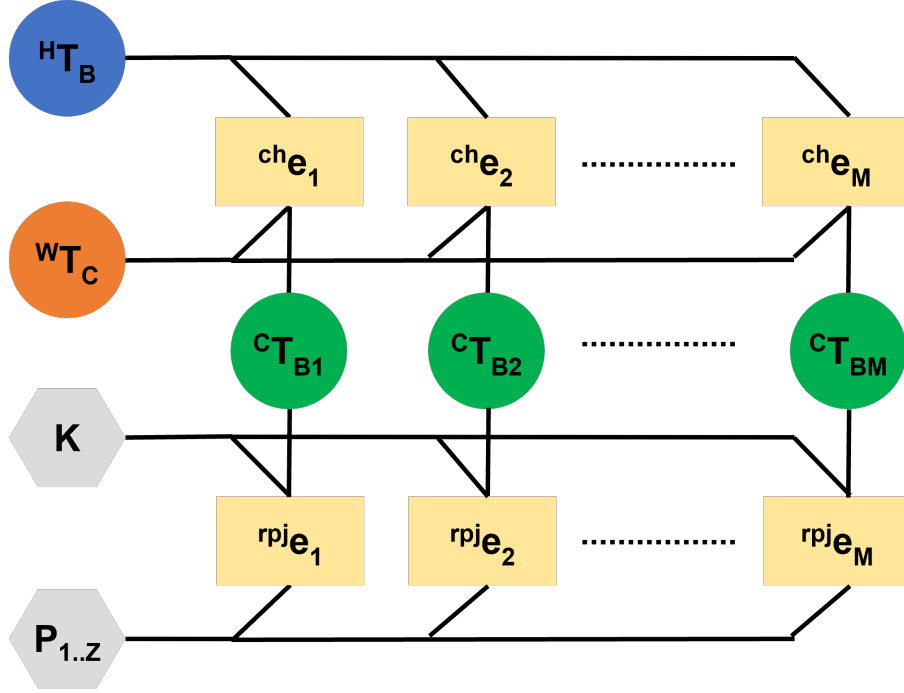


Figure 3.3: Graphical representation of the optimization problem for eye-on-base single camera setups. The representation contains circles, rectangles and hexagons that respectively represent the estimated variables, error functions, and fixed parameters. The error functions are defined following the equations 3.3 and 3.4.

where $\mathbf{h}(\cdot)$ is the measurement function and $\mathbf{z}_i$ is the i -th observation and $\boxminus$ is the difference operator for the manifold domain.

The first type of edge that is going to be presented is the one that enforces the equality :

$$^W T_H \cdot ^H T_B = ^W T_C \cdot ^C T_B, \quad (3.2)$$

that is one of the classic constraints for eye-on-base calibration [72]. And its corresponding error is defined as follows:

$$^{ch} e_i = t2v([\mathbf{H} T_W]_i \cdot ^W T_C \cdot [^C T_B]_i \cdot ^B T_H), \quad (3.3)$$

where the observation $\mathbf{z}_i = [^W T_H]_i^{-1}$ is the i -th pose of the end-effector in the world. The remaining transformations are the ones that need to be estimated, while the $t2v(\cdot)$ is a function that converts the transformation to the corresponding minimal manifold representation in order to prevent optimization degradation due to over parametrization.

The second type of edge instead enforces the reprojection error constraint:

$${}^{rpj}\mathbf{e}_i = \begin{bmatrix} \pi([\mathbf{C}\mathbf{T}_\mathbf{B}]_i \cdot {}^B\mathbf{P}_1) - [\mathbf{u}_1]_i \\ \vdots \\ \pi([\mathbf{C}\mathbf{T}_\mathbf{B}]_i \cdot {}^B\mathbf{P}_j) - [\mathbf{u}_j]_i \\ \vdots \\ \pi([\mathbf{C}\mathbf{T}_\mathbf{B}]_i \cdot {}^B\mathbf{P}_Z) - [\mathbf{u}_Z]_i \end{bmatrix}, \quad (3.4)$$

where the observation $\mathbf{z}_i = [\mathbf{u}_1 \dots \mathbf{u}_j \dots \mathbf{u}_Z]_i$ are vectors of tuples $\mathbf{u}_j = (u_x, u_y)_j$ that are the pixel coordinates of the j -th detected corner of the checkerboard at the i -th observation and π is the projection function of the camera.

$$\pi({}^C\mathbf{P}_j) = K \cdot {}^C\mathbf{P}_j \quad (3.5)$$

The projection function $\pi(\cdot)$ maps the point ${}^C\mathbf{P}_j$ in the camera reference frame, to the image space using the intrinsics camera matrix K , which characterizes the specific camera used.

The optimization aims to minimize the following function:

$$\min \sum_{i=1}^M (\lambda \|{}^{rpj}\mathbf{e}_i\|^2 + \|{}^{ch}\mathbf{e}_i\|^2), \quad (3.6)$$

where λ is a weighting factor that accounts for the non-homogeneity of the two error terms. The minimization is carried out by using the Levenberg-Marquardt [73] algorithm. Our method proved to be robust also if provided with an identity transformation as initial guess, showing a rather large convergence basin. This means that the system can be correctly calibrated even if we do not provide a good initial guess, which could be difficult to compute.

3.3.2 EYE-ON-BASE CALIBRATION FOR MULTI-CAMERA

If the robot is observed by multiple static cameras, one could repeat the above algorithm several times. On the other hand, running an algorithm that performs single-camera calibration N times, with N the number of cameras that are present in the workspace, is a sub-optimal solution because, in the optimization, the cross-observations between the different cameras are not factored in the problem.

Given N cameras in the workspace, as shown in Fig. 3.1, there is the need to introduce a little more notation:

- ${}^C\mathbf{T}_{\mathbf{C}_a}$: Isometry representing the pose of the a -th camera in the d -th camera reference frame;
- ${}^C\mathbf{T}_\mathbf{B}$: Isometry representing the pose of the checkerboard in the a -th camera reference frame;

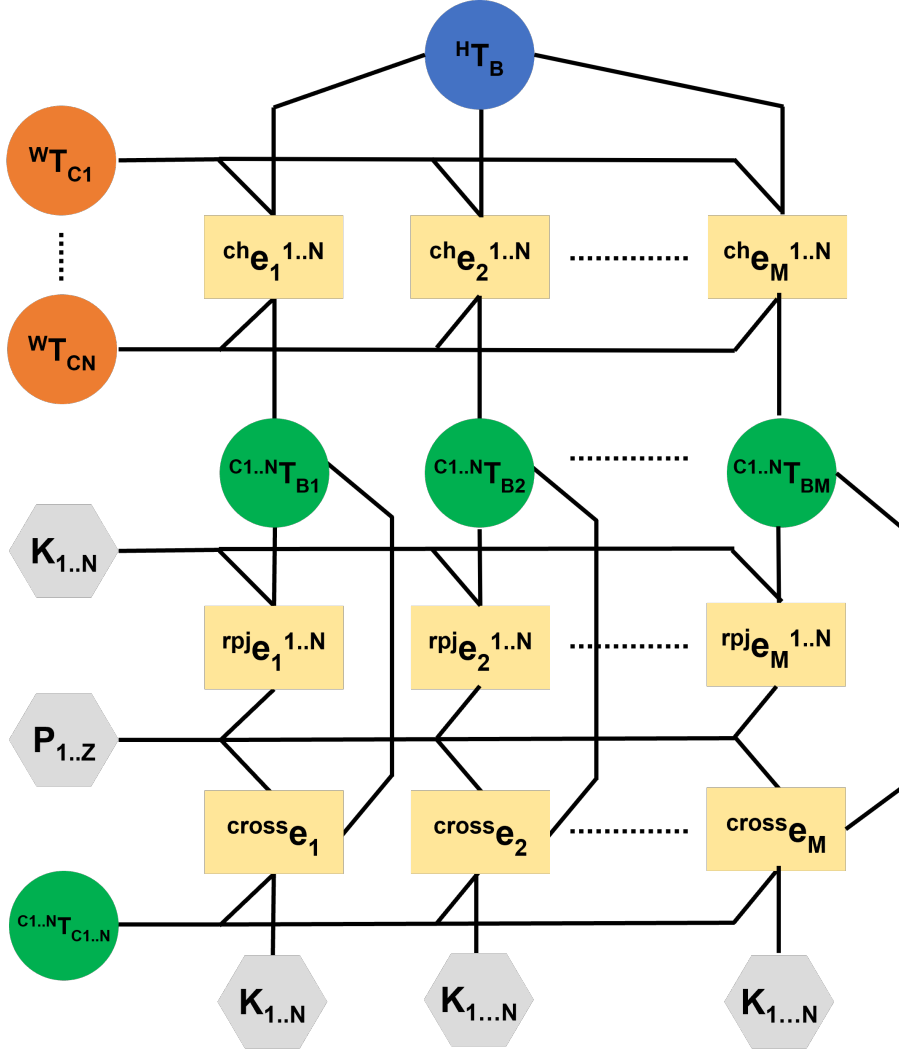


Figure 3.4: Evolution of the graph representation in multi cameras setups. The figure reports the error terms of Eq. (3.7) and Eq. (3.8) that are the stack of the errors in Eq. (3.3) and Eq. (3.4), respectively. Moreover, the graph includes an additional error defined in Eq. (3.11) which enforces constraints between the cameras with overlapping field of view.

- $W^T_{C_a}$: Isometry representing the pose of the a -th camera in the world reference frame.

The starting point for the multi-camera model is simply stacking the single-camera model N times as the number of the cameras. In Fig. 3.4 some of the vertices and edges have been grouped together in order to facilitate the graphical representation.

The error term $^{ch}e_i^{1..N}$ enforces all the chained equalities of the system :

$$^{ch}e_i^{1..N} = [^{ch}e_i^1 \dots ^{ch}e_i^a \dots ^{ch}e_i^N]^T, \quad (3.7)$$

where ${}^{ch}\mathbf{e}_i^a$ represents the chained equality for the a -th camera at the i -th time step. The error term relative to the reprojection error evolves into:

$${}^{rpj}\mathbf{e}_i^{1..N} = [{}^{rpj}\mathbf{e}_i^1 \dots {}^{rpj}\mathbf{e}_i^a \dots {}^{rpj}\mathbf{e}_i^N]^T, \quad (3.8)$$

where ${}^{rpj}\mathbf{e}_i^a$ is the reprojection error term relative to the camera a -th at the i -th time step. Up to this point, we have simply stacked the single-camera graph N times one on top of the other, in fact the derivation of the new error terms was straightforward. The additional error term that is added to the graph model is the one that represents the cross-observation between cameras, from which information about their relative spatial relation can be extracted. Cross-observations occur when more cameras are able to see the checkerboard at the same time step i . In order to add it in an automatic manner to the optimization, it is needed to build a co-visibility matrix $\mathbf{X}_i$. At each time step i , each cell of this matrix is represented by $\mathbf{X}_i(a, d)$, where a and d are two different cameras (i.e., $a \neq d$). The cell contains a binary value that encodes whether or not the checkerboard is in the field of view of both cameras a and d at the i -th time step:

$$\mathbf{X}_i(a, d) = \begin{cases} 0, & \text{if } a = d \text{ no cross-observation} \\ 1, & \text{otherwise} \end{cases} \quad (3.9)$$

Thus, using the co-visibility matrix we are able to enforce the cross-observation constraints only where they are actually aiding the optimization.

To capture the spatial relation between cameras the transformation ${}^{C_d}\mathbf{T}_{C_a}$ is added as a state variable that needs to be estimated.

The new error term that models the cross-observation is defined as follows:

$${}^{cross}\mathbf{e}_{iad} = \begin{bmatrix} \pi_a({}^{C_a}\mathbf{T}_{C_d} \cdot [{}^{C_d}\mathbf{T}_B]_i \cdot \mathbf{P}_1^B) - [\mathbf{u}_1^a]_i \\ \vdots \\ \pi_a({}^{C_a}\mathbf{T}_{C_d} \cdot [{}^{C_d}\mathbf{T}_B]_i \cdot \mathbf{P}_j^B) - [\mathbf{u}_j^a]_i \\ \vdots \\ \pi_a({}^{C_a}\mathbf{T}_{C_d} \cdot [{}^{C_d}\mathbf{T}_B]_i \cdot \mathbf{P}_Z^B) - [\mathbf{u}_Z^a]_i \end{bmatrix}, \quad (3.10)$$

where the indices a and d refer to the cameras and $[\mathbf{u}_j^a]_i$ represents the j -th corner of the checkerboard detected by the camera a at the i -th time step. This error accounts for the current estimate of the relative transformation between cameras. Checkerboard points are projected into the camera c_a using the camera c_d as starting point.

To simplify the notation all the error terms at time step i will be marginalized:

$${}^{cross}\mathbf{e}_i = \sum_{a=1}^N \sum_{d=1}^N \|{}^{cross}\mathbf{e}_{iad}\|^2 \cdot \mathbf{X}_i(a, d), \text{ with } a \neq d. \quad (3.11)$$

Fig. 3.4 shows the complete graphical model that includes these final constraints. The resulting function that needs to be minimized is the following:

$$\min \sum_{i=1}^M (\lambda_1 \|\mathbf{e}_i^{rpj}\|^2 + \|\mathbf{e}_i^{ch}\|^2 + \lambda_2^{cross} \mathbf{e}_i) \quad (3.12)$$

The effect that the cross-observations have on the optimization is to make it more robust against noise, and it will be effectively shown in the following section.

The proposed algorithm has been implemented using the g2o [74] framework. The weighting factors λ_1 and λ_2 have been experimentally tuned and set to the values $[10^{-6}, 10^{-3}]$, respectively, and kept fixed during all experiments.

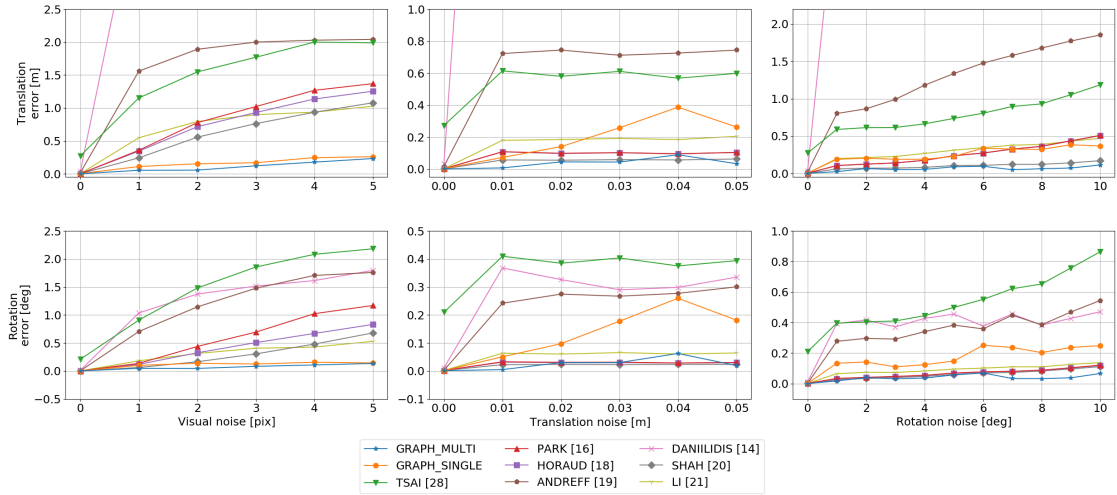


Figure 3.5: Result of the simulated evaluation. The proposed approach, both in *single*- and *multi*-camera settings has been tested against Tsai [75], Park [63], Horaud [65], Andreff [66], Daniilidis [61], Shah [67] and Li [68].

3.4 EVALUATION

The robustness and estimation accuracy of the proposed method is demonstrated by testing it under visual and geometric perturbations. In particular, we both added noise to the detection of the checkerboard corners, thus simulating wrong detections of the calibration pattern (e.g., when using low-resolution cameras), and we also added noise to the robot poses $[\mathbf{H}\mathbf{T}_{\mathbf{W}}]_i$ used for calibrating.

The proposed approach has been tested against the following methods (i.e., implementation from OpenCV[†]): [75, 63, 65, 66, 61, 67] and [68]. In the graphs and in the table we name

[†]http://docs.opencv.org/4.5.4/d9/d0c/group__calib3d.html

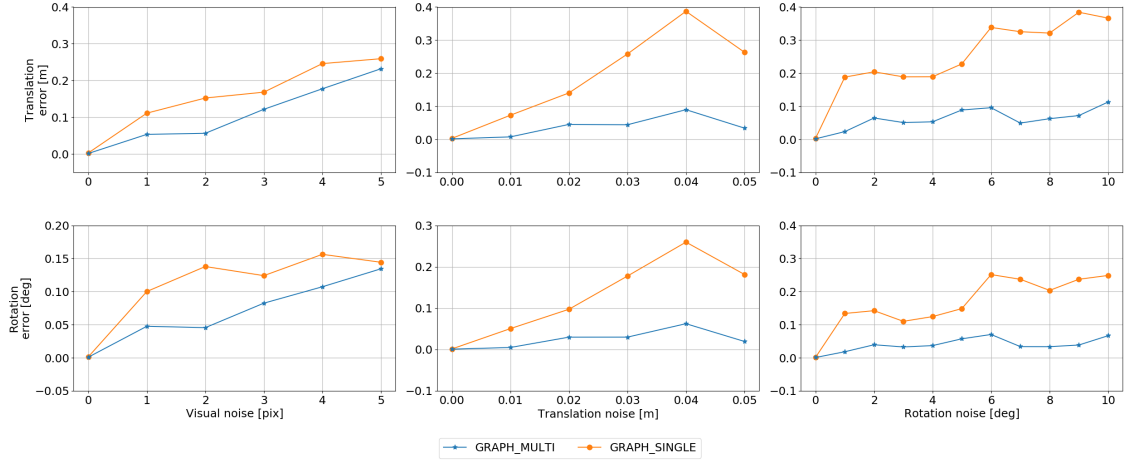


Figure 3.6: More in depth focus of the simulated evaluation regarding the proposed method only in *single*- and *multi-camera* settings.

them by reporting the first author of each one. The alternative methods have been tested by calibrating every single camera independently and then averaging the errors. This experimental methodology has been adopted because the alternative approaches are not directly usable for multi-camera calibration. The evaluation also reports an in-depth focus on the comparison of our multi-camera calibration method against the repetition of the proposed single-camera calibration applied to the individual cameras, this is done to demonstrate the robustness and improved accuracy of the multi-camera calibration framework.

3.4.1 EXPERIMENTS WITH SYNTHETIC DATA

Using the same evaluation protocol proposed in [56, 76], our method has been tested, under simulated conditions, using three different sources of noise: visual, translation, and rotation. This evaluation protocol aims to test the robustness of the calibration under perturbed visual conditions (i.e., introducing noise in the detection of the corners of the calibration pattern), and it is also used to test the optimization accuracy and convergence capabilities when additional geometric noise is added to the robot poses.

We collected simulated data using the *Gazebo*[‡] simulator, in which a simulated work cell has been implemented. The proposed work cell is composed of an industrial manipulator, mounting the calibration pattern, surrounded by five external cameras as in typical multi-camera eye-on-base setups. A set of 150 robot poses has been selected by randomly sampling the robot workspace to uniformly cover each camera field of view.

The results of this evaluation are reported in Fig. 3.5 and Fig. 3.6. In particular, in Fig. 3.5 the plots show the performance of all the calibration methods, while in Fig. 3.6 only the results

[‡]<https://gazebo.org/home>

Table 3.1: Evaluation of the reprojection errors using the real setup data. All values are in pixels.

Camera \ Error [pix]	Tsai [75]	Park [63]	Horaud [65]	Andreff [66]	Daniilidis [61]	Shah [67]	LI [68]	GRAPH MULTI
Camera 1	443.76	0.52	0.43	1.41	196253	0.50	0.61	0.34
Camera 2	0.42	0.35	0.35	1.98	0.54	0.44	0.72	0.28
Camera 3	148.00	26.29	25.06	165429	38.90	97.88	417.00	4.42
Average	197.69	9.05	8.62	55144	65431	32.94	139.00	1.68

of our single- and multi-camera graph-based calibration approaches are reported. The results show how our proposed method achieves good results in terms of translation and rotational errors even in strong perturbation conditions, demonstrating that it is more robust compared to other methods.

The proposed eye-on-base single-camera calibration (i.e., GRAPH_SINGLE in Fig. 3.5 and Fig. 3.6) generally outperforms all the other tested approaches. Moreover, our multi-camera calibration hand-eye algorithm (i.e. GRAPH_MULTI in Fig. 3.5 and Fig. 3.6) is the best performing approach, in particular when high levels of noise are present in the calibration data.

3.4.2 EXPERIMENTS ON A REAL SETUP

The proposed algorithm has also been validated in a real robotic environment. The setup consisted of a Franka Emika Panda[§] robot manipulator and three Kinect2 cameras (we use only the RGB images) placed around the robot, as shown in Fig. 3.7 and Fig. 3.8. Unlike the simulated evaluation, where ground truth is available, in the real environment the absolute pose of each camera is not known a priori. Thus, to perform the test in the real setup, the reprojection error has been considered as a metric. In Tab. 3.1 the results of all approaches are reported.

Our approach is capable of correctly calibrating all cameras in the setup, also in challenging lighting condition, e.g., for cameras #1 and #3 (see Fig. 3.7). Most of the tested methods were unable to converge to a low reprojection error solution, while our method provided the lowest error by a large margin.

3.5 CONCLUSION

This chapter presents a PG solution to the hand-eye calibration problem that extends previous state-of-the-art work [56] in two different ways: i) by formulating the solution to *eye-on-base* setups with one camera; ii) by extending the *eye-on-base* calibration to multi-camera setups. To

[§]<https://www.franka.de/>

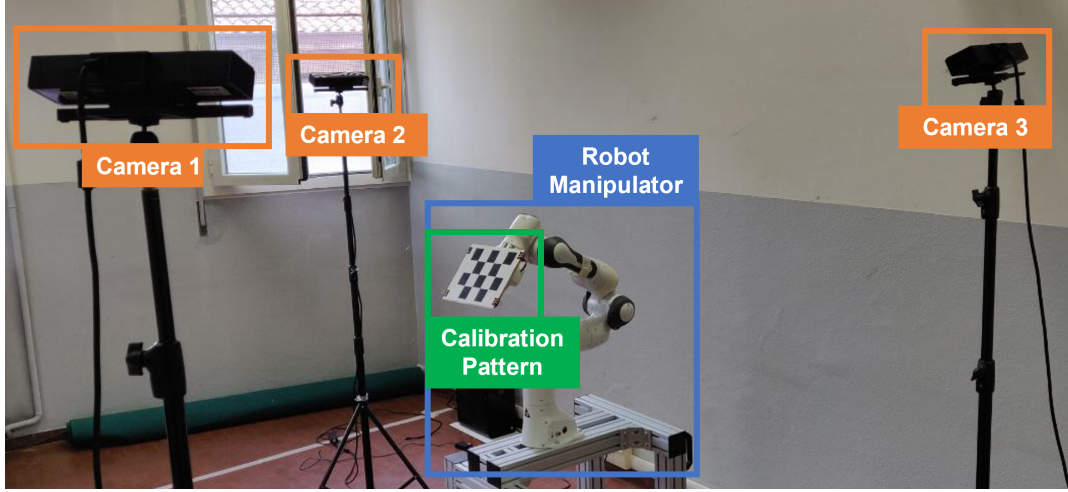


Figure 3.7: Front view of the real setup where the proposed calibration method has been tested.

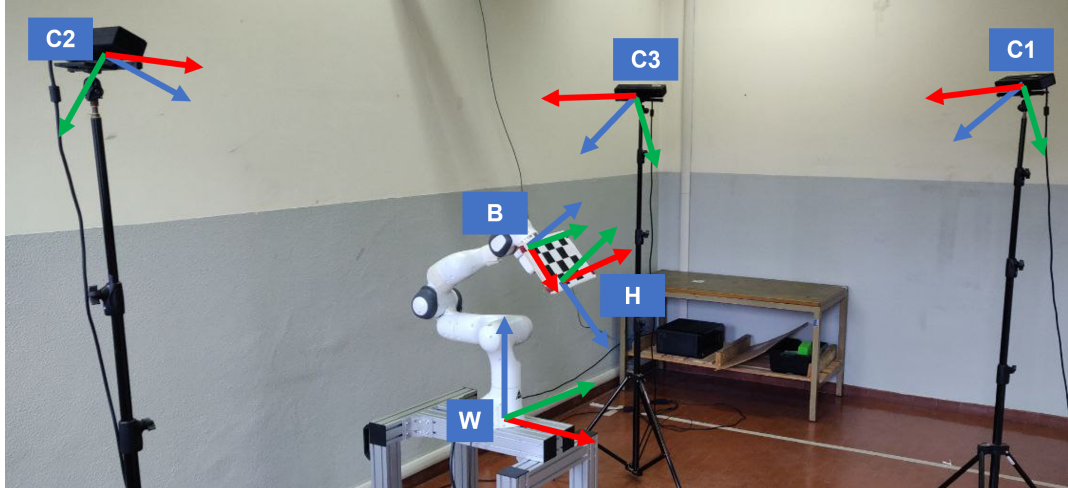


Figure 3.8: Rear view of the real setup. Main reference frames are depicted.

validate the proposed system, we used the same experimental protocol proposed in [56] and [76].

The proposed approach has shown to be robust when the calibration data are affected by visual and geometric errors. Compared to traditional solutions, the implemented pose-graph optimization achieves better accuracy in terms of geometric errors (i.e., translation and rotation).

The proposed approach also proves to be robust and accurate when tested in real hand-eye calibration benchmarks, where absolute ground truth is not available and accuracy is measured in terms of reprojection error.

It should be noticed that the proposed approach, as in the other hand-eye calibration meth-

ods used for the experiments, does not take into account the influence of dynamic effects given by gravity or joint and link flexibility. This may affect the accuracy and precision of robot manipulators and must be taken into account with proper dynamic calibration techniques.

The authors believe that this kind of calibration can be formulated with the same graph-based structure as in the proposed hand-eye calibration approach, and it may be considered as future work in this field to reach a more general calibration procedure. Moreover, the proposed graph-based formulation can be further extended in future research to address more complex problems such as calibration of *multi-robot-multi-camera* setups.

4

IPC: Incremental Probabilistic Consensus-based Consistent Set Maximization for SLAM Backends

In SLAM problems, Pose Graph Optimization has become the de facto standard for refining an initial estimate of a set of poses using pairwise relative measurements. These measurements fall into two categories: sequential and non-sequential. Non-sequential measurements are typically generated by Place Recognition modules, which are prone to failure in highly aliased and unstructured environments. Throughout this chapter, we refer to these failures as outliers. While PGO effectively manages noisy measurements, it lacks inherent mechanisms to handle outliers—where even a single erroneous constraint can lead to catastrophic and meaningless results. While there have been methods proposed to extend PGO and handle large numbers of outliers, these approaches are sensitive to initialization, dataset-dependent parameters, and are restricted to offline usage. This chapter therefore presents our proposed approach: Incremental Probabilistic Consensus (IPC). IPC approximates the solution to the combinatorial problem of finding the maximally consistent set of measurements. The solution is found in an incremental fashion, by evaluating the consistency of each loop closure measurement through a consensus-based procedure. Consensus is determined from a subset of the global problem, where all previously integrated inlier measurements have veto power.

4.1 MOTIVATION

SLAM is the problem of computing the robot’s pose and a globally consistent map of the environment at the same time given a sequence of sensor readings. It is universally recognized as a fundamental problem for making a robot truly autonomous, with a vast literature spread over almost 40 years [4, 77]. Visual SLAM [78, 79] refers to those SLAM systems that use

cameras as the main input sensors, for example, to calculate the robot's motions [80] (often referred to as *visual odometry*), to extract landmarks or objects of interest [81], and to detect previously visited locations [82] (often referred to as *visual place recognition* and used to solve the *loop closure detection* problem). Typical SLAM implementations basically include two layers: the *front-end* which processes raw sensory data (e.g. encoder readings, images, LiDAR scans, etc....) to estimate useful information such as the relative displacements between two sensor readings [15, 83, 14, 84]; the *back-end* which fuses the data produced by the front-end to obtain a globally consistent map and trajectory [85, 86, 87]. In Visual SLAM systems [88, 89, 90, 91], the front-end usually includes the visual odometry and the loop closure detection modules [92, 93, 94, 3]. In most recent SLAM systems, the back-end includes an optimization module that solves a Maximum a Posteriori (MAP) estimation problem, where the unknown variables and their measured spatial relationships are organized inside a *factor graph* [4]. In this graphical representation, nodes denote the robot poses (in Visual SLAM they are often associated with a subset of relevant images called *keyframes*), and landmarks, while edges (i.e., factors) between them encode spatial relationships provided by the front-end. To increase the scalability and efficiency of the back-end, the factor graph is often reduced to include only the robot poses as nodes, where each factor denotes a constraint between a couple of poses. This framework is called *Pose Graph Optimization (PGO)* [33], and is widely used in most recent Visual SLAM systems.

Spatial constraints are usually classified into *odometry constraints*, which connect consecutive poses (e.g., keyframes), and *loop closure constraints*, which connect non-consecutive poses and are triggered by the loop closure detection module. Most SLAM and Visual SLAM frame-

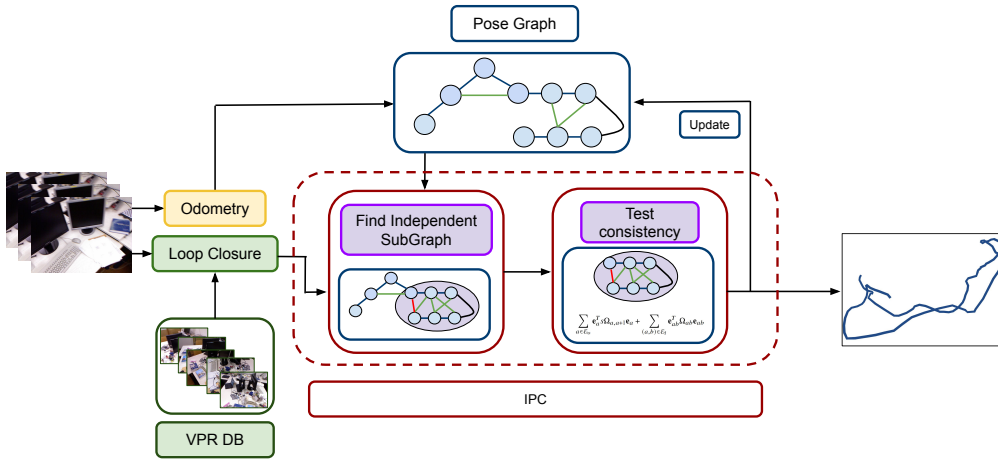


Figure 4.1: Overview of the IPC framework integrated into a Visual SLAM system. Visual odometry constraints generated by the front-end are directly incorporated into the PG, while loop closure constraints from the VPR module undergo verification using the consistency set on their independent subgraph. If a measurement is deemed consistent, the trajectory is updated accordingly.

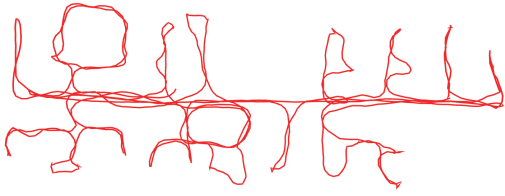
works (e.g., [88, 89, 90, 95]) assume that loop closure constraints, once detected and validated by the detector (e.g., [96]), are affected by noise but are free of outliers. This forces these systems to use very conservative thresholds in the detector, so as to minimize the possibility of introducing incorrect constraints, with the consequence of discarding potentially useful loop closure detections. On the other hand, in challenging environments characterized by a high perceptual aliasing (i.e., in which different locations share similar visual appearance), the integration of loop closure outliers is a concrete option that could lead to irreversible divergences in optimization.

PGO generally utilizes Non-Linear Least Squares (NLLS) techniques to solve a non-convex MAP estimation problem, where the solution is the nodes configuration that best aligns with the problem’s constraints. Common libraries used for solving the PGO problem include g2o [86], GTSAM [85], and Ceres [87]. NLLS techniques are particularly vulnerable to poor initializations and to outliers. In complex environments, aliasing effects make it unrealistic to expect the front-end to always avoid mistakes when relying on isolated observations. Consequently, it should be the back-end that takes charge of proactively dealing with erroneous measurements, thanks to its ability to access the complete measurement history.

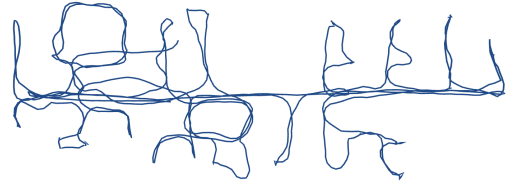
In this chapter, we address the challenge of managing outliers that can infiltrate the optimization system. Addressing this problem is crucial, as even a single outlier can destabilize the entire system and compromise the validity of the solution. Existing solutions that address this problem build upon classic NLLS solvers by employing methods such as RANSAC [97], adding robust error functions [98], evaluating constraint consistency [99, 100, 101], or gradually transitioning from a more convex problem formulation to reintroduce non-linearity. Our contribution falls in the Consistent Set Maximization category. The proposed approach tries to approximate the solution to the combinatorial problem of finding the maximally consistent set of measurements in an incremental fashion. While the majority of solutions are designed for offline usage, our approach is able to achieve online performances, making it suitable for use during missions rather than solely in post-processing. For each incoming loop closure measurement, we find and optimize only a subgraph for which this constraint has an active influence. The solution to this subproblem is tested by using previously accepted measurements, each one with veto power, to decide whether to accept the new measurement. If accepted, also the subgraph optimization is propagated to the rest of the graph.

The main contributions of this chapter are the following:

- (i) IPC: An online, consensus-based approach able to incrementally build a good approximation of the maximally consistent set of loop closure measurements;
- (ii) An exhaustive evaluation of the proposed method. The evaluation is performed in standard datasets used for benchmarking state-of-the-art outlier rejection solutions;
- (iii) An open-source implementation of IPC available at: <https://github.com/EmilioOlivastri/IPC>



(a) Ground truth trajectory



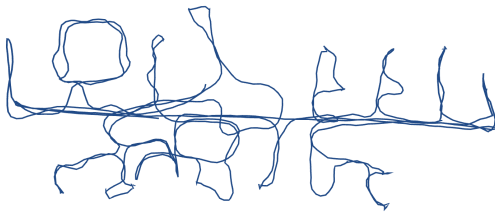
(b) IPC



(c) ADAPT[98]



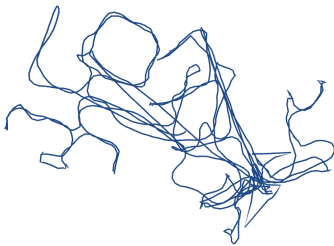
(d) DCS[102]



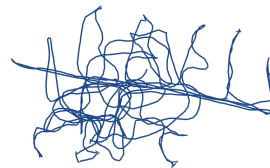
(e) GM[103]



(f) GNC[5]



(g) PCM[99]



(h) MAXMIX[104]

Figure 4.2: Trajectory estimated by the different methods on the FR079 dataset corrupted with 70% of outliers.

4.2 RELATED WORK

Interest in Robust Pose Graph Optimization (RPGO) has increased in recent years, as optimization is a delicate process, and even a few outliers can significantly impact the final solution. To mitigate this, various techniques for RPGO have been proposed throughout the literature. One of the earliest approaches for trying to mitigate the effects of outliers on the optimization process are the M-estimators [103]. These robust cost functions replace the standard quadratic error to prevent the optimizer from minimizing errors associated with outliers. M-estimators, such as Huber, Cauchy loss, German-McClure, and Truncated-Least-Squares, reduce or nullify the influence of errors exceeding a maximum acceptable threshold. The region within which errors are considered is referred to as the “trust region”. M-estimators inject additional non-linearity into the problem and are sensitive to the threshold defining the trust region. Selecting an appropriate threshold and M-estimator is problem-specific and often non-trivial. In [98] a new robust loss function called (ADAPT) was introduced. The ADAPT loss is a general loss capable of representing the various robust cost functions, selecting the preferred one through a control parameter. This approach eliminates the need to manually choose the most suitable cost function, as the optimizer treats the control parameter as a latent variable and automatically determines the appropriate level of robustness.

Graduated Non-Convexity (GNC) [5] is an iterative framework that solves a non-convex problem by first addressing its more convex approximations. Initially, a convex approximation of the original problem is solved, and with each iteration, non-convexity is gradually introduced until the original problem is fully restored. The core idea behind this approach is that the solution to a simpler version of the problem serves as an effective starting point for solving the more complex problem. GNC filters out outliers more effectively than the direct application of an M-estimator. This arises from GNC’s ability to better manage the non-linearity introduced by robust cost functions, albeit at the expense of increased convergence time.

In [105], the authors aim to address both the non-linearity introduced by robust cost functions and the task of manually selecting a cost function with AEROS. AEROS combines the GNC paradigm with the general ADAPT loss [98]. In [106], a version of GNC suited for incremental solvers, such as iSAM2 [107] is proposed. The use of the Scale-Invariant Graduated (SIG) Kernel enables a dramatic reduction in the number of iteration steps required to achieve convergence.

A widely adopted alternative for solving RPGO involves defining an augmented problem that integrates outlier removal as part of the cost function to be minimized. In [108], “switchable” variables were introduced to the classic PGO formulation. These variables are used to determine whether a measurement is an inlier or an outlier, automatically removing outliers from the optimization process. In [102], an equivalent formulation is presented that eliminates the need for these additional variables, thereby reducing convergence time. Instead of using vari-

ables to model the presence of outliers, [104] employed a multi-modal Gaussian distribution. The optimizer selects the most promising Gaussian at each iteration by using the max-mixture distribution as the underlying model.

As noted in [109], the inlier or outlier status of measurements is not always directly observable. Consequently, joint consistency among measurements has emerged as a key criterion for RPGO. Instead of explicitly distinguishing inliers from outliers, methods in this category prioritize identifying the largest subset of mutually consistent measurements, leveraging the inherent consistency between inliers. However, determining the maximally consistent set of measurements poses a problem of combinatorial complexity. This has pushed the research community to develop effective heuristics and approximations for the problem. In [101] it is proposed an exact method that is not exponential in most cases, but it is still not viable for real-time or online applications. The proposed method Joint Compatibility Branch and Bound (JCBB) leverages the Branch and Bound technique in conjunction with the joint compatibility test between measurements.

In [99], it was proposed the Pairwise Consistency Maximization (PCM) method. This approach approximates the joint consistency check using pairwise consistency and employs heuristic Max-Clique algorithms to compute the largest set of pairwise consistent measurements. ROBIN [110] introduces and adopts a new criterion, to effectively build the compatibility graph, called invariance. An invariant assesses the consistency between measurements independently of the problem’s state variable estimates. Defining an effective invariant requires exploiting the inherent priors and properties of the specific problem at hand. In [100] different registration problems are addressed, and for each one specific geometric-based invariants are defined. The proposed method uses the projection gradient ascent with a back-tracking line search to maintain low execution time while still being able to compute the largest set of consistent measurements.

A more heuristic approach is proposed in [111], where measurements are temporally grouped into clusters, and these clusters are used for testing both intra- and inter-cluster consistency. The largest set of consistent measurements is approximated to all measurements that pass both tests. [112] builds on the previous approach by introducing an additional term designed to mitigate inconsistencies caused by noisy odometry measurements. However, it comes at the cost of increased execution time. In [113], instead of temporally grouping measurements, all measurements that constrain the same subproblem are used to test for consistency.

4.3 METHOD

We expect the robot to be equipped with (i) an ego-motion estimation front-end (e.g., wheel odometry and/or visual-inertial odometry) that provides the rigid transformations between

consecutive poses $\mathbf{x}_a, \mathbf{x}_{a+1} \in \text{SE}(2)$ (or $\text{SE}(3)$) over time ; (ii) a place recognition module that can detect medium- and long-term previously seen locations and estimate the transformation between the related previous pose $\mathbf{x}_a$ and the current one $\mathbf{x}_b$ (i.e., a *loop closure measurement* or *constraint*). Given the sequence of odometry and loop closure measurements, our method aims to find the maximum consistent set of measurements while estimating the whole trajectory followed by the robot.

4.3.1 ASSUMPTIONS AND HEURISTICS

Our method builds upon the following assumptions:

A1: *Odometry measurements are inliers.* The quality of the odometry can change based on the front-end that is used, but as stated in [109], they are usually not corrupted by outliers. These measurements are reliable constraints that can be used to enforce maintaining the local shape of the trajectory.

A2: *Prevalence of inliers.* Given the inherent aliasing effect in the environment, it is normal for the place recognition module to produce a certain percentage of outliers. On the other hand, it is reasonable to expect that the majority of loop closure measurements are inliers. Furthermore considering **A1** and the fact that the frequency of odometry measurements is generally higher than the one of loop closure measurements, the prevalence of inlier measurements is guaranteed.

As generally done in robust optimization, in IPC we apply the following heuristics:

H1: *Start from a simpler version of the problem, for which its solution will be meaningful for the more complex version of the problem.* As stated in [114] and later in [20], a simpler version of a problem exhibits a broader and more stable convergence basin, thus ensuring better convergence. Consequently, the solution obtained for the reduced problem can be used as the initial guess for the full problem, also improving the convergence of the latter.

H2: *The nature of the compatibility test is to reject as many outliers as possible while preserving the inliers.* The distinction between outliers and inliers is not always observable as stated in [109], so instead of trying to detect which measurements are outliers, consistency can be verified using Eq. (4.4).

4.3.2 PROBLEM FORMULATION

A pose graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ uses nodes in $\mathcal{V}$ to represent the discretized poses $\mathbf{x}_i$ of the robot (i.e., the followed trajectory) while the edges $(a, b) \in \mathcal{E}$ that connect the nodes represent the spatial relations (i.e., the measurements, or constraints) between them. The full state can be represented using the vector $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top]^\top$, where the subscript defines the temporal order and n is the total number of poses. The goal of PGO is to find a solution $\mathbf{x}^*$ that minimizes

the following objective function:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{(a,b) \in \mathcal{E}} \mathbf{e}(\mathbf{x}_a, \mathbf{x}_b)^T \Omega_{ab} \mathbf{e}(\mathbf{x}_a, \mathbf{x}_b) \quad (4.1)$$

where Ω_{ab} is the information matrix associated with the measurement collected between $\mathbf{x}_a$ and $\mathbf{x}_b$ and the error is defined as follows:

$$\mathbf{e}(\mathbf{x}_a, \mathbf{x}_b) = \mathbf{h}(\mathbf{x}_a, \mathbf{x}_b) \boxminus \mathbf{z}_{ab}. \quad (4.2)$$

where $\mathbf{h}(\mathbf{x}_a, \mathbf{x}_b)$ is a function that computes the rigid transformation between the two poses, $\mathbf{z}_{ab}$ is the odometry or loop closure measurement, and $\boxminus$ is the difference operator for the manifold domain. We use the notation $\mathbf{x}_i$ to both refer to a robot pose and the corresponding node, and $\mathbf{e}(\mathbf{x}_a, \mathbf{x}_b)$ to both refer to an error term and the corresponding edge of $\mathcal{G}$. Following [111], we partition the edges in two sets, $\mathcal{E} = \mathcal{E}_o \cup \mathcal{E}_l$, where $\mathcal{E}_o = \{(a, b) \in \mathcal{E} \mid a < b \wedge b = a + 1\}$ is the set of edges associated with the odometry measurements, and $\mathcal{E}_l = \{(a, b) \in \mathcal{E} \mid a < b \wedge b \neq a + 1\}$ is the set of edges associated with the loop closure measurements.

To make the notation more fluid the following shortcuts are used:

- $\mathbf{e}_a = \mathbf{e}(\mathbf{x}_a, \mathbf{x}_{a+1})$ for the odometry errors;
- $\mathbf{e}_{ab} = \mathbf{e}(\mathbf{x}_a, \mathbf{x}_b)$ for the loop closure.

It allows us to rewrite Eq. (4.1) as:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{(a,a+1) \in \mathcal{E}_o} \mathbf{e}_a^T \Omega_{a,a+1} \mathbf{e}_a + \sum_{(a,b) \in \mathcal{E}_l} \mathbf{e}_{ab}^T \Omega_{ab} \mathbf{e}_{ab} \quad (4.3)$$

4.3.3 INDEPENDENT SUBGRAPHS AND SOLUTION PROPAGATION

In the simplest case, a loop closure measurement (e.g., the green edge between $\mathbf{x}_1$, and $\mathbf{x}_5$ in Fig. 4.3) creates in the pose graph $\mathcal{G}$ a loop without edge crossing or internal loops; all nodes that compose this loop form a subgraph $\mathcal{G}'$ (in the example, $\mathcal{G}'$ is composed of the set $\{\mathbf{x}_1, \dots, \mathbf{x}_5\}$, highlighted in purple) that can be optimized independently from the other nodes in $\mathcal{G} \setminus \mathcal{G}'$. A subgraph defined in this way always defines a time interval that includes all nodes, and related edges, from index a to index b , with $a < b$ and $b - a > 1$. More formally, we can define $\mathcal{G}' = \mathcal{G}(a, b) = (\mathcal{V}(a, b), \mathcal{E}(a, b))$ where $\mathcal{V}(a, b) = \{\mathbf{x}_i \in \mathcal{V} \mid a \leq i \leq b\}$ and $\mathcal{E}(a, b) = \{\mathbf{e}_{ij} \in \mathcal{E} \mid a \leq i \wedge j \leq b\}$. If no other node $\mathbf{x}_k \in \mathcal{G} \setminus \mathcal{G}(a, b)$ is able to generate a loop with any of $\mathbf{x}_i \in \mathcal{G}(a, b)$, we define $\mathcal{G}(a, b)$ as an *independent subgraph* $\mathcal{G}^I(a, b)$ that can be optimized independently from all the other nodes (see Fig. 4.3). The result of this optimization can be *propagated* to all other nodes $\mathbf{x}_k \in \mathcal{G} \setminus \mathcal{G}'$ simply by applying the estimated transformation updates of the node in $\mathcal{G}'$ they are connected to. For example, after optimizing the subgraph

highlighted in purple in Fig. 4.3, the result of this optimization can be propagated to the nodes $\mathbf{x}_k \in \mathcal{G}(6, 13)$ by adding the transformation update applied to $\mathbf{x}_5$.

Independent subgraphs can contain multiple loops and cross edges: for example in Fig. 4.3 the arrival of a loop closure between node 7 and node 13 causes the fusion between the two previous independent subgraphs represented by the light green ovals.

4.3.4 THE IPC ALGORITHM

The IPC algorithm is fundamentally very simple and exploits and updates a set CnS (called **Consensus Set**) of loop closure measurements that have been accepted in previous iterations. The following steps briefly outline the IPC algorithm, where we report in square brackets the used assumptions and heuristics (see Sec. 4.3.1):

1. Integrates all odometry measurements and generates a new node when necessary (for example, if the distance to the previous node is above a certain threshold) [$\mathbf{A1}$];
2. For each incoming loop closure measurement $\mathbf{z}'$, find the (minimal) independent subgraph $\mathcal{G}'$ (see Sec. 4.3.3) that contains such constraint by considering odometry constraints and only the loop closure constraint in CnS collected so far;
3. Solve the PGO problem only for the independent subgraph $\mathcal{G}'$ [$\mathbf{H1}$];
4. Accept and include $\mathbf{z}'$ in CnS only if *all* measurements in $\mathcal{G}'$ are consistent (i.e., *agree*, see below) with the new solution obtained in the previous point [$\mathbf{A1}, \mathbf{A2}, \mathbf{H1}, \mathbf{H2}$];

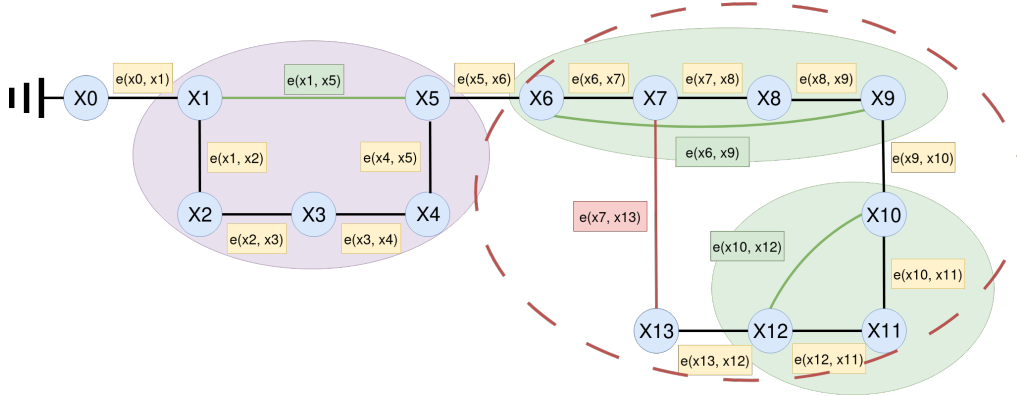


Figure 4.3: An example of pose graph with nodes represented by light blue circles. Each error term is associated with the corresponding edge. The coordinate system is usually fixed in the first node $\mathbf{x}_0$. Light green and purple ovals represent examples of simple independent subgraphs (i.e., they only include one loop) while the dotted red oval represents more complex independent subgraphs, that include both multiple loops with crossing edges and internal loops.

Algorithm 5: IPC

```

1  $CnS, \mathcal{E}_o, \mathcal{E}_l, C \leftarrow \emptyset$ 
2  $\mathcal{V} \leftarrow \{\mathbf{x}_0\}$ 
3 upon receiving  $\mathbf{z}_{ij}$  then
    // odometry measurement
4   if  $j = i + 1$  then
5      $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i \boxplus \mathbf{z}_{i,i+1}$  // integrate motion
6      $\mathbf{e}_i \leftarrow \text{edge}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{z}_{i,i+1}, \Omega_{i,i+1})$ 
7      $\mathcal{E}_o \leftarrow \mathcal{E}_o \cup \{\mathbf{e}_i\}$  // update edges
8      $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{x}_{i+1}\}$  // update variables
    // loop closure measurement
9   else
    // find initial  $\mathcal{G}^I$ 
10     $\mathbf{e}_{ij} \leftarrow \text{edge}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}, \Omega_{ij})$ 
11     $\mathcal{V}^I, \mathcal{E}^I \leftarrow \text{find } \mathcal{G}_{ij}^I$ 
    // find cross edges until independent
12     $a \leftarrow i$ 
13    while  $\exists \mathbf{x}_k \in \mathcal{G} \setminus \mathcal{G}^I \mid \exists \mathbf{e}_{kg} \in CnS, a < g < j$  do
14       $\mathcal{E}^I \leftarrow \mathcal{E}^I \cup \{\mathbf{e}_k, \dots, \mathbf{e}_a\}$ 
15       $\mathcal{E}^I \leftarrow \mathcal{E}^I \cup \{\mathbf{e}_{kg}\}$ 
16       $\mathcal{V}^I \leftarrow \mathcal{V}^I \cup \{\mathbf{x}_k, \dots, \mathbf{x}_{a-1}\}$ 
17       $a \leftarrow k$ 
18     $\mathcal{E}_{so}, \mathcal{E}_{sl} \leftarrow \mathcal{E}^I$ 
19    solve Eq. (4.6) // PGO
20    if  $\mathbf{e}_{ab}^T \Omega_{ab} \mathbf{e}_{ab} < \chi_{\alpha, \delta}^2, \forall \mathbf{e}_{ab} \in \mathcal{E}^I$  then
21       $CnS \leftarrow CnS \cup \{\mathbf{e}_{ij}\}$ 
22       $\mathcal{E}_l \leftarrow \mathcal{E}_l \cup \{\mathbf{e}_{ij}\}$ 
23      propagate solution to  $\mathbf{x}_k \in \mathcal{V} \setminus \mathcal{V}^I$  as described in Sec. 4.3.3
24    else
25      restore( $\mathcal{V}^I$ )

```

5. If $\mathbf{z}'$ has been accepted, propagate the obtained solution to the remaining nodes as described in Sec. 4.3.3.

To check whether a measurement agrees with a PGO solution, we exploit the χ^2 test[101]:

$$\mathbf{e}_{ab}^T \Omega_{ab} \mathbf{e}_{ab} < \chi_{\alpha, \delta}^2, \forall \mathbf{e}_{ab} \in \mathcal{E}' \quad (4.4)$$

where α is the confidence parameter and δ the degrees of freedom of the error and $\mathcal{E}'$ is the edgset of $\mathcal{G}'$. The error term is computed here by using as robot pose the solution obtained by

the PGO carried out on $\mathcal{G}'$. If one of the tests fails, the new loop closure measurement $\mathbf{z}'$ is *not* accepted, hence not included in CnS .

We report the pseudocode of our method in Alg. 5, which is described step by step below. For the sake of simplicity, we assume here to create a new node for each new odometry measurement.

As soon as a new measurement $\mathbf{z}_{ij}$ is received by the system, the first operation to carry out is to identify if it is an odometry or a loop closure measurement. If an odometry measurement is received, then $\mathbf{z}_{ij} = \mathbf{z}_{i,i+1}$ (line 4), a new variable $\mathbf{x}_{i+1}$ is initialized integrating the motion estimate $\mathbf{z}_{i,i+1}$ from $\mathbf{x}_i$ (line 5), and then added to $\mathcal{V}$. The final operation is to update the set $\mathcal{E}_o$ with the new edge $\mathbf{e}_i$ relative to the measurement $\mathbf{z}_{i,i+1}$ (line 7).

In the scenario a loop closure measurement $\mathbf{z}_{ij}$ is received (line 9), its relative edge $\mathbf{e}_{ij}$ is created. Then, $\mathbf{e}_{ij}$ is used to identify the minimal independent subgraph $\mathcal{G}^I(a, j)$ (line 11-17) such that:

$$\nexists \mathbf{x}_k \in \mathcal{G} \setminus \mathcal{G}^I(a, j) \mid \exists \mathbf{e}_{kg} \in CnS \wedge a < g < j \quad (4.5)$$

Once, $\mathcal{G}^I(a, j)$ has been identified, its edges are decomposed in the following manner $\mathcal{E} = \mathcal{E}_o \cup \mathcal{E}_l$. This step is fundamental because being the odometry measurements the “trustworthy” ones, a scaling factor $s > 1$ is applied to their information matrix to increase the importance of minimizing their error during the optimization using Eq. (4.6) (line 19):

$$\sum_{a \in \mathcal{E}_o} \mathbf{e}_a^T s \Omega_{a,a+1} \mathbf{e}_a + \sum_{(a,b) \in \mathcal{E}_l} \mathbf{e}_{ab}^T \Omega_{ab} \mathbf{e}_{ab} \quad (4.6)$$

The parameter s allows the solver to prioritize maintaining the local shape of the trajectory, actively reducing the acceptance rate of outliers. The scaling factor is fixed and does not play a role in the optimization’s complexity, thus avoiding an increase in the problem’s complexity, as opposed to the switchable constraints [108] where additional extra variables were added. Not satisfying Eq. (4.4) implies that the optimizer could not find a configuration of the nodes for which the new edge is consistent with the measurements in $\mathcal{G}^I$. Therefore, each element of $\mathcal{V}^I$ is restored to its state prior to the optimization. Instead, if Eq. (4.4) is satisfied, a consistent configuration is found leading to the inclusion of the loop closure to the CnS (line 21). The current solution obtained for the nodes $\mathcal{V}^I$ is then propagated to the nodes that did not participate in the optimization $\mathcal{V} \setminus \mathcal{V}^I$ (line 23).

4.4 EVALUATION

We implemented the proposed method (called IPC in all experiments) in C++ building upon the g2o[86] framework, exploiting the Dog-Leg optimization steps and setting $s = 3$ for all experiments. We compared IPC with 7 recent alternative, state-of-the-art approaches, reported below along with the acronym used in the plots and the used implementation. Most of these methods aim to mitigate the anomalous problem in loop closure detection. For Max-Mixture

method [104] (MAXMIX in the plots) in the experiment we used the implementation provided in OpenSLAM*. We implemented inside g2o the Adaptive Kernel [98] robust kernel (ADAPT in the plots). For Pairwise consistent measurement set maximization method [99] (PCM in the plots), Kimera’s implementation is used[115]. The remaining methods are provided by the gtsam library[85]: Graduated Non-Convexity [5], Dynamic Covariance Scaling [102], Huber and German-McClure [103], will be referred to in the plots respectively as GNC, DCS, HUBER, and GM. For all methods we kept all their parameters constant, reporting for each one of them the results obtained with the best set of parameters. Experiments were carried out in a cluster PC using a single Intel(R) Xeon(R) Gold 5220 CPU.

4.4.1 DATASETS

The datasets used for evaluation are currently the standard for evaluating the performances of back-end optimizers and are described in [116] [117]. They can be divided into two categories, synthetic and real. The Intel, Mit, Csail, Freiburg Building (Fig. 4.2), and Freiburg University Hospital are real datasets, while M3500 is synthetic. In both cases, the datasets are outlier-free. The loop closure edges in the unaltered datasets were used as the ground truth for the inliers. Outliers were added, using the Vertigo package [118], as a percentage of the total inliers present in the dataset in order to have balanced results.

4.4.2 METRICS

The **Precision**, **Recall**, and **F1 score** metrics are used in the evaluation, the latter defined from the former:

$$Prec = \frac{TP}{TP + FP}, Rec = \frac{TP}{TP + FN}, F1 = 2 * \frac{Prec * Rec}{Prec + Rec}$$

where **TP**, **FP**, **TN** **FN** stand for a loop closure that is a true and false positive and true and false negative, respectively. Finally, the solution obtained using SE-Sync [119], which is a fast non-minimal solver for PGO, on the outlier free version of the dataset was used as ground truth trajectory for the evaluation of both **Absolute Trajectory Error** (ATE) and **Relative Pose Error** (RPE)[120], that will be used to measure the accuracy of estimated trajectory.

4.4.3 EXPERIMENTS

The introduced outliers range from 10% to 100% of the ground truth inliers, with increments of ten percent. To ensure fair and robust testing, we generated ten distinct trajectories for each outlier percentage. A percentage of 100% means that the number of outliers is equal to the number of inliers. As done in [106], for methods that don’t explicitly classify measurements, an inlier is defined as a measurement having a χ^2 error less than χ_{th}^2 with $\alpha_{th} = 0.95$. For all

*<https://openslam-org.github.io/>

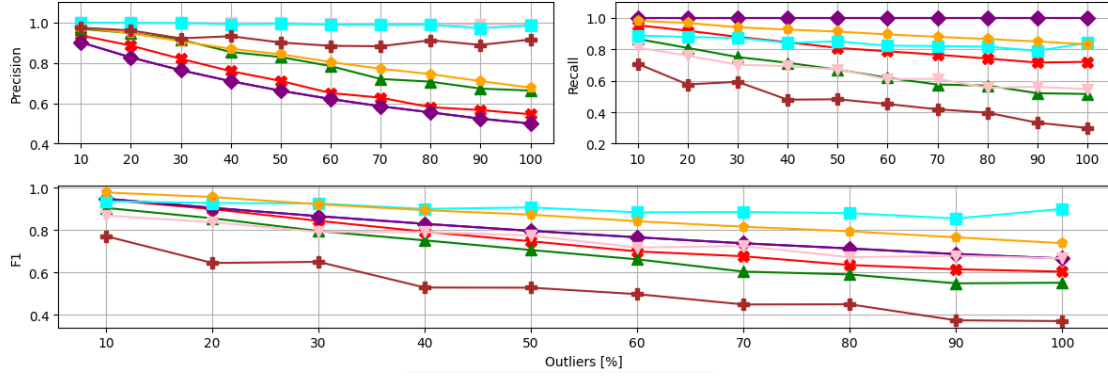


Figure 4.4: Top left, top right, and bottom images are respectively the precision, the recall, and F1 score performances of the methods for varying outlier percentages. Legend of the graphs: HUBER ($\blacktriangle$), GM ($\bullet$), IPC ($\square$), DCS ($\blacklozenge$), MAXMIX ($\blacktriangledown$), ADAPT ($\blacklozenge$), PCM ($\blackplus$), GNC ($\blackcross$). GM is completely covered by DCS in both precision and recall plots, while MAXMIX is nearly entirely covered by IPC in the precision plot.

Method	50%			100%		
	F1 $\uparrow$	RPE [m] $\downarrow$	TIME [s] $\downarrow$	F1 $\uparrow$	RPE [m] $\downarrow$	TIME [s] $\downarrow$
GNC	0.747	32.20	155.86	0.604	35.53	349.04
ADAPT	0.873	1.867	5457.07	0.738	1.856	12576.27
MAXMIX	0.77	0.324	70.99	0.66	0.43	137.29
DCS	0.796	26.12	4.58	0.66	37.684	8.28
GM	0.795	23.64	7.07	0.65	38.33	9.59
HUBER	0.706	21.44	10.04	0.55	27.32	8.29
PCM	0.53	25.01	8289.04	0.371	28.54	12655.41
IPC	0.91	0.05	321.37	0.89	0.068	442.733

Table 4.1: F1, RPE, and runtime values from Fig. 4.4 and in Fig. 4.5 for 50% and 100% of outliers.

methods, the different CnS s are evaluated at the end of the optimization of the full problem. One small note to take into account is that results for both GNC and PCM were taken after a fixed number of iterations. It was done due to their large convergence time and the exhaustive amount of testing that was carried out for each method.

Fig. 4.4 shows the performances for precision, recall, and F1 score averaged on all the used datasets. The precision is one of the most crucial parameters for a correct termination for the optimization. Its value should ideally approach one since even minor outlier contributions have the potential to severely compromise the trajectory estimate. MAXMIX and IPC are the methods that reject most of the outliers across the different datasets, closely followed by PCM. Conversely, for GNC, DCS, and GM when the percentage of outliers exceeds 50%, the precision drops below 0.5, strongly reducing the probability that the solver reaches a good solution.

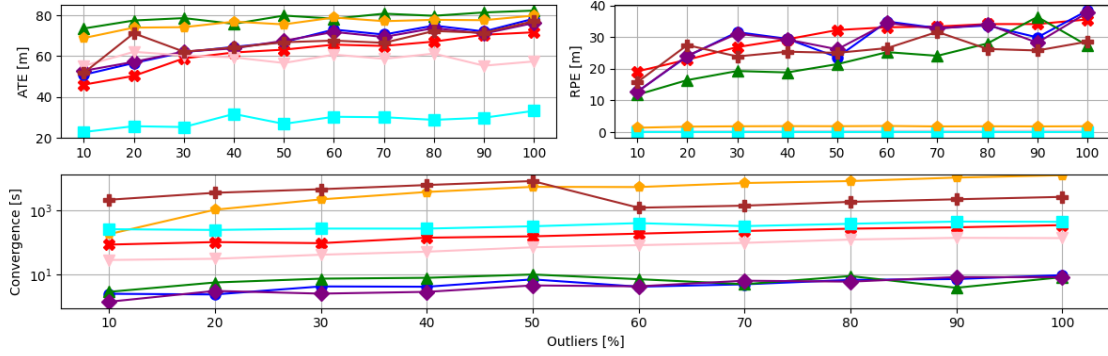


Figure 4.5: Top left, top right, and bottom images are respectively the ATE, the RPE, and convergence time of the methods for varying outlier percentages. Legend of the graphs: HUBER (▲), GM (●), IPC (□), DCS (◆), MAXMIX (▽), ADAPT (○), PCM (⊕), GNC (✕). The MAXMIX plot for the RPE metrics is totally covered by the ADAPT plot.

Simply having high precision is not sufficient, because having poor recall performances will negatively impact the elimination of the accumulated drift due to the high number of inliers discarded, also resulting in poor trajectory estimation. Still, the acceptable bound is not as strict as the one used for precision. As shown in Fig. 4.4, all methods have an almost constant recall trend as the percentage of outliers increases. GM and DCS demonstrate the highest recall rates, followed by ADAPT and IPC which provide a consistent recall of over 80%. Nonetheless, IPC outperforms all these methods in the accuracy metric by a large margin, as shown in the RPE column in Tab. 4.1. The F1 score, which relates both precision and recall together, gives a global view of the actual methods' performance. It can be seen that the methods report a similar trend with respect to the precision. It is an expected behavior given the quasi-constant performance of the recall. ADAPT slightly dominates for smaller percentages of outliers, while IPC is able to keep stable and reliable results for all the different percentages, resulting in having the best performances for higher percentages of outliers.

Fig. 4.5 shows respectively the convergence time, the ATE, and the RPE metrics. We can see that for the ATE almost all methods have similar results, with IPC clearly overtaking the other methods. ATE highlights how globally brittle the optimization is even in the presence of a small number of outliers, while RPE measures the local consistency of the estimated trajectories. IPC, MAXMIX, and ADAPT have the best RPE of the different methods.

To compare the offline methods' time performance with IPC, it is used the time needed to validate all the loop closure constraints provided in the different datasets. As expected, it is not able to compete with the robust estimators and DCS in terms of speed, but it provides comparable results to more complex methods such as GNC and MAXMIX, even if they were specifically designed to work offline.

In Tab. 4.2 for each dataset is reported the Average Convergence Time per Constraint (AC-TxC), that is the average convergence time on the subproblem identified by a loop closure constraint. The worst results are the ones for the M3500, because the majority of outliers captured

Table 4.2: IPC’s ACTxC for each dataset as the outlier percentage varies. The reported results are expressed in seconds.

Dataset	20%	40%	60%	80%	100%
MIT	0.053	0.050	0.050	0.050	0.044
INTEL	0.056	0.051	0.058	0.054	0.056
M3500	0.577	0.542	0.717	0.601	0.623
CSAIL	0.043	0.042	0.043	0.042	0.043
FRH	0.098	0.095	0.100	0.098	0.109
FR079	0.042	0.042	0.043	0.041	0.043

a vast section of the trajectory, leading to predominantly solving subproblems of high dimensionality. Also, in case a false positive of such dimension is accepted, it would negatively impact the clusters’ size and the future constraints’ convergence time. Still, the results obtained fit the online deployment of the method.

4.5 CONCLUSIONS

We presented IPC, a consensus-based method to incrementally build the maximum coherent set of measurements. IPC demonstrates comparable or better performance with respect to recent state-of-the-art approaches while providing online capabilities. In contrast to most methods, IPC shows great generalization capabilities by showing consistent results across the different datasets without changing its parameters, and its only control parameter s is fairly simple to configure. Currently, IPC’s primary limitation is its inability to rectify past erroneous decisions. Therefore, a possible future direction could involve investigating the specific contexts where IPC is more prone to making errors and also integrating an offline method on top of it that revises and possibly discards the previously integrated measurements, in order to gain the benefits of both methodologies.

5

TACO: A Test and Check Framework for Robust Pose Graph Optimization

Incremental Probabilistic Consensus (IPC), as presented in Ch. 4, is a greedy online method that incrementally approximates the combinatorial problem of finding the largest set of consistent measurements. In Pose Graph Optimization, this corresponds to identifying the maximal coherent set of loop closures. Due to IPC’s greedy nature, if a mistake is made when evaluating the consistency of the measurement, the optimization will not be able to recover. In this chapter, we address IPC’s limitation by introducing TACO (Test And Check Optimization). TACO is a Robust Pose Graph Optimization (RPGO) framework which introduces two complementary modules: a short-term horizon module that incrementally tests loop closure constraints while maintaining online processing capabilities; and a long-term horizon module that revises the integrated constraints to correct potential errors. The former extends IPC for consistent loop closure testing by focusing on a subset of the most influential constraints, thereby improving efficiency; the latter integrates the Switchable Constraints (SC) framework within a randomized voting scheme, enhancing convergence rates while reducing computation time.

5.1 MOTIVATION

Incremental Probabilistic Consensus (IPC) was our initial step toward developing a Robust Pose Graph Optimization (RPGO) framework. To enable online performance, IPC approximates the combinatorial problem of identifying the largest set of consistent loop closure measurements. Its decision-making process relies on a consistency test, which assumes that introducing an outlier will create a previously non-existent inconsistency in the system. This assumption holds as long as the PG, formed by both odometry and loop closure measurements, is sufficiently connected—meaning that, in NLLS terms, the underlying optimization problem is well-constrained. However, in real-world applications, this condition is not always met.

A practical example of this assumption not holding arises in long, open-loop trajectories where no places have been revisited, and pose estimation relies solely on odometry, leading to significant drift accumulation. If an outlier is introduced in such a scenario by the Place Recognition (PR) module, the weakly connected PG increases the risk of integrating incorrect constraints. Since IPC assumes that all measurements except the one under test are inliers, incorporating a single outlier can systematically bias future loop closure decisions. To mitigate these failure cases, it is crucial to implement mechanisms that enable the system to recover from past mistakes and avoid propagating erroneous constraints. One potential solution is leveraging offline RPGO techniques as error-recovery procedures. Existing approaches address the impact of outliers by extending traditional NLLS solvers through various strategies, including RANSAC-based outlier rejection [97], robust error functions [98], and constraint consistency evaluation [99, 100, 101]. Some methods further improve optimization stability by gradually refining the problem formulation, initially favoring more convex approximations before reintroducing nonlinearity. Despite their effectiveness, these methods come with two key limitations. First, their offline nature restricts them to post-processing, preventing their direct application in real-time scenarios where immediate corrections are necessary. Second, some techniques significantly increase the computational complexity of PGO, making convergence more sensitive to initialization and requiring highly accurate initial estimates. These limitations underscore the need for a more versatile approach that can handle both online robustness and long-term error

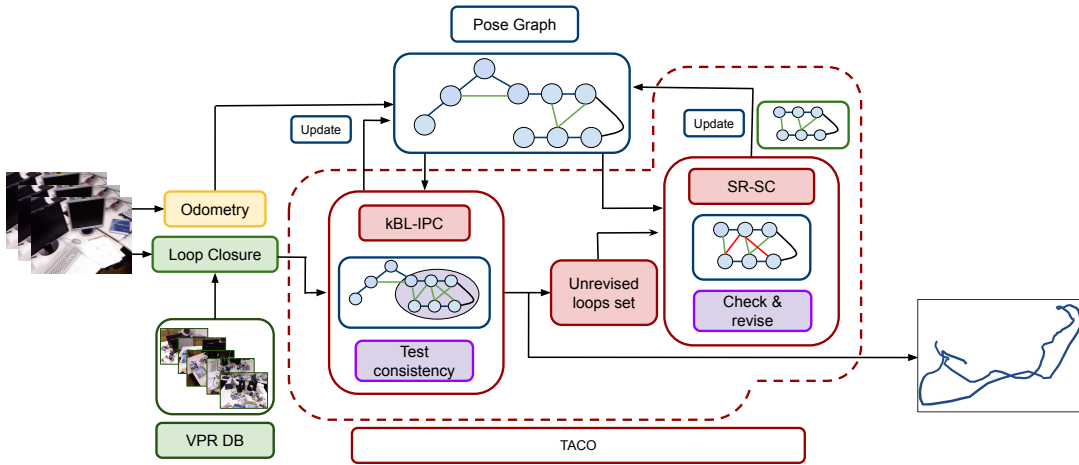


Figure 5.1: Overview of the TACO framework used inside a Visual SLAM system. The front-end generates both sequential constraints (visual odometry) and loop closures from a sequence of images (VPR DB stands for visual place recognition database). kBL-IPC tests the consistency of candidate loop closures against their best loop matches. The output of kBL-IPC provides an online estimate of the trajectory (bottom right in blue). Each candidate loop closure, classified as an inlier or outlier, is then added to a set called *unrevised set*. Once such a set reaches size R , SR-SC will double-check the loop closures and update the original pose graph, along with the estimates of the related variables.

correction. To address these challenges, this chapter presents a novel RPGO framework that operates across two distinct time horizons. The first component is a short-term module designed for online decision-making, ensuring robustness while maintaining efficiency. The second component is a long-term module responsible for error correction, continuously refining past decisions to rectify mistakes introduced by the short-term module. By integrating these two complementary processes, the proposed framework provides a more resilient and adaptable solution to robust pose graph optimization. In this chapter, we introduce TACO, short for Test And Check Optimization, a robust framework for PGO consisting of two complementary modules, each addressing a distinct time horizon and defined by the acronym’s keywords “test” and “check” (see Fig. 5.1 for an overview):

- kBL-IPC: k-Best Loops IPC, a short-time horizon module that incrementally *tests* the measured constraints (i.e., the loop closures) to ensure robustness while maintaining online processing capabilities;
- SR-SC: Selective Random Switchable Constraints, a long-time horizon module that *checks* the integrated constraints, aiming to correct potential errors introduced by the short-term module.

The short-time horizon module is an extension of the previously presented IPC (Ch 4), that falls under the category of Consistent Set Maximization methods. We enhance this module by significantly increasing its speed while maintaining comparable performance. This is achieved by focusing on a limited number of the most influential (“best”) loop closure detection constraints to assess the current loop closure candidate. The long-time horizon module leverages the Switchable Constraints (SC) framework [108] within a randomized voting scheme applied to a subset of the original problem to perform error recovery while reducing the temporal overhead. This approach provides a retrospective revision of the online decisions made by kBL-IPC, potentially running in parallel for efficiency. These two modules can operate independently and possibly can also be integrated within other frameworks. The primary contributions of this chapter are as follows:

- (i) kBL-IPC, an extension of an existing approach that strategically selects the top constraints for consistent loop closure testing, enabling users to balance speed and reliability;
- (ii) SR-SC, an error recovery strategy that embeds the switchable constraints [108] into a randomized voting scheme, enhancing convergence rates while minimizing computation time;
- (iii) An analysis of the various failure modes that can occur when integrating outliers, highlighting why traditional metrics used for evaluating outlier-free SLAM solutions are insufficient for assessing the robustness of PGO systems, and identifying the most suitable metrics for evaluating Robust PGO (RPGO);

- (iv) A comprehensive evaluation of the proposed method alongside state-of-the-art algorithms, using both standard 2D datasets and Visual SLAM datasets from vision-based sources, for testing outlier rejection, which demonstrates that our approach achieves overall superior results;
- (v) An open-source implementation of TACO, made publicly available at: <https://github.com/EmilioOlivastri/TACO>

Beyond developing a robust RPGO framework, an equally critical challenge lies in effectively evaluating its robustness and reliability. Since RPGO is still an emerging research area, its assessment has largely relied on conventional SLAM metrics, such as Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). While these metrics offer valuable insights into trajectory accuracy, they inherently assume that all integrated measurements are inliers, making them inadequate for evaluating RPGO performance in the presence of outliers. As a result, they fail to capture the resilience of the optimization process under challenging conditions. To overcome this limitation, it is essential to adopt evaluation methodologies that account for the robustness of RPGO against outlier contamination. In this chapter, we examine the shortcomings of standard SLAM metrics in this context and explain why they fail to provide a complete picture of RPGO performance. Furthermore, we introduce alternative assessment strategies designed to offer a more comprehensive evaluation, facilitating the selection of an appropriate RPGO framework and its corresponding parameters.

5.2 PRELIMINARIES

As previously introduced, the front-end of SLAM systems includes a motion estimation module that provides a set of odometry measurements (i.e., constraints) as relative rigid-body transformation estimates $\mathbf{z}_{a,a+1} \in \text{SE}(3)$ between consecutive poses (or keyframes in Visual SLAM) $\mathbf{x}_a, \mathbf{x}_{a+1} \in \text{SE}(3)$ defined at discrete times a and b . These poses represent absolute transformations in the map's coordinate system. The front-end is also responsible for detecting previously visited locations over medium- and long- terms using a PR module, and estimating the relative rigid-body transformation $\mathbf{z}_{a,b} \in \text{SE}(3)$ between the associated previous pose, $\mathbf{x}_a$, and the current pose, $\mathbf{x}_b$ (i.e., a loop closure measurement or constraint). The back-end leverages all the constraints generated by the front-end to refine the estimates of the poses $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ through an optimization process. The PGO approach uses a graph where nodes represent the robot's poses while edges capture their spatial relationships. The next section introduces the core concepts of PGO.

5.2.1 POSE GRAPH OPTIMIZATION FORMULATION

A pose graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ uses nodes in $\mathcal{V}$ to represent the discretized poses $\mathbf{x}_i$ of the robot (i.e., the followed trajectory) while the edges $(a, b) \in \mathcal{E}$ that connect the nodes represent mea-

surements, or constraints (in the rest of the chapter we use these terms interchangeably). The full state can be represented using the vector $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top]^\top$, where the subscript defines the temporal order and n is the total number of poses. The goal of PGO is to find a solution $\mathbf{x}^*$ that minimizes the following objective function:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{(a,b) \in \mathcal{E}} \mathbf{e}(\mathbf{x}_a, \mathbf{x}_b)^T \Omega_{a,b} \mathbf{e}(\mathbf{x}_a, \mathbf{x}_b) \quad (5.1)$$

where Ω_{ab} is the information matrix associated with the measurement between $\mathbf{x}_a$ and $\mathbf{x}_b$. Its error is defined as follows:

$$\mathbf{e}(\mathbf{x}_a, \mathbf{x}_b) = \mathbf{h}(\mathbf{x}_a, \mathbf{x}_b) \boxminus \mathbf{z}_{a,b} \quad (5.2)$$

where $\mathbf{h}(\mathbf{x}_a, \mathbf{x}_b)$ is a function that computes the rigid transformation between the two poses, $\mathbf{z}_{a,b}$ is the odometry or loop closure measurement, and $\boxminus$ is the difference operator for the manifold domain. The solution to Eq. (5.1) is obtained by solving a NLLS Optimization problem using minimization algorithms such as Gauss-Newton (GN) [36], Levenberg-Marquardt (LM) [37], or Powell's Dog Leg (PDL) [38]. We use the notation $\mathbf{x}_i$ to both refer to a robot pose and the corresponding node, and $\mathbf{e}_{a,b}(\mathbf{x}_a, \mathbf{x}_b)$ to both refer to an error term and the corresponding edge $(a, b) \in \mathcal{E}$. Following [111], we partition the edges in two sets, $\mathcal{E} = \mathcal{E}_o \cup \mathcal{E}_l$, where $\mathcal{E}_o = \{(a, b) \in \mathcal{E} \mid a < b \mid b = a + 1\}$ is the set of edges associated with the odometry measurements, and $\mathcal{E}_l = \{(a, b) \in \mathcal{E} \mid a < b \mid b \neq a + 1\}$ is the set of edges associated with the loop closure measurements. A robust pose graph optimization approach should be able to handle outlier loop closure measurements, so we further distinguish among inlier $^i\mathcal{E}_l$ and outlier $^o\mathcal{E}_l$ loop closure measurements, $\mathcal{E}_l = ^i\mathcal{E}_l \cup ^o\mathcal{E}_l$.

To make the notation more fluid the following abbreviations are used:

- $\mathbf{e}_{a,a+1} = \mathbf{e}(\mathbf{x}_a, \mathbf{x}_{a+1})$ for the odometry errors;
- $\mathbf{e}_{a,b} = \mathbf{e}(\mathbf{x}_a, \mathbf{x}_b)$ for the loop closures.

We can rewrite Eq. (5.1) as:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{(a,a+1) \in \mathcal{E}_o} \mathbf{e}_{a,a+1}^T \Omega_{a,a+1} \mathbf{e}_{a,a+1} + \sum_{(a,b) \in ^i\mathcal{E}_l} \mathbf{e}_{a,b}^T \Omega_{a,b} \mathbf{e}_{a,b} \quad (5.3)$$

where in an ideal case only inlier loop closure measurements should be taken into account. The goal of our approach is to minimize the probability of integrating outliers during optimization.

5.3 METHOD

This section details the two main components of the proposed TACO framework (Fig. 5.1). TACO operates across two distinct time horizons. The first module conducts online tests to quickly assess the consistency of incoming loop closures. It is based on a modified version of

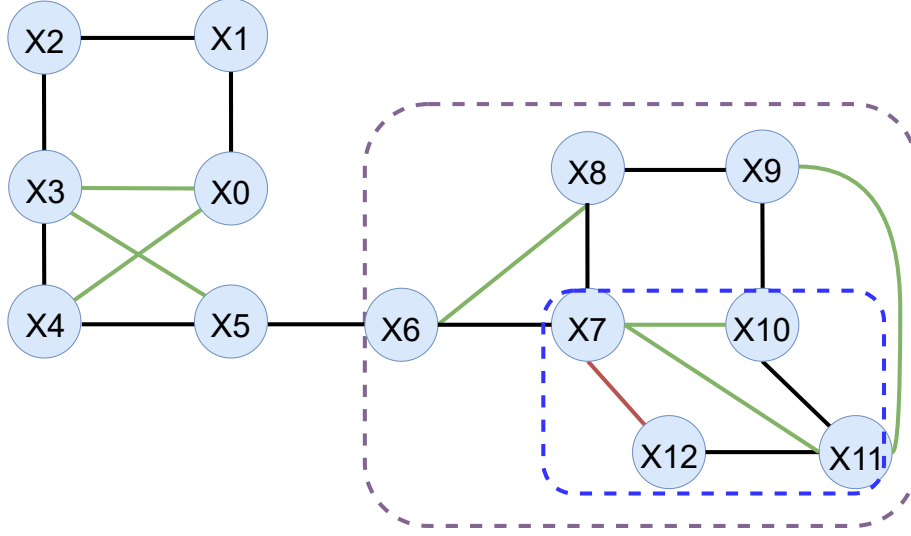


Figure 5.2: The figure illustrates a pose graph where the odometry constraints are represented by black edges, accepted loop closures by green edges, and the candidate loop closure by a red edge. In this example, the candidate edge is $e(\mathbf{x}_7, \mathbf{x}_{12})$. The larger purple rectangle highlights the minimal independent subgraph selected by IPC (Ch 4), while the smaller blue rectangle denotes the subgraph chosen by kBL-IPC, where in this case $k = 2$.

IPC (Ch 4), which reduces the computational time for a single consistency test while maintaining comparable performances. The second module performs a slower, more thorough evaluation of previously assessed constraints, allowing for the removal of incorrectly integrated outliers and the reintegration of initially rejected inliers. It is based on the SC framework [108], integrated into a randomized voting scheme. After R evaluations of loop constraints from the first module, the error recovery procedure is triggered to prevent any negative impact on future decisions. We adopt here a constructive approach, initially presenting the basic methods we rely on and their limitations, followed by a detailed discussion of our modifications and contributions.

5.3.1 IPC: INCREMENTAL PROBABILISTIC CONSENSUS

IPC (Ch 4) is an online method that incrementally approximates the combinatorial problem of finding the largest set of consistent measurements, which in PGO usually corresponds to identifying the maximal coherent set of loop closures. Instead of focusing on outlier detection, IPC incrementally tests for consistency between measurements, since the classification between inliers/outliers is generally not observable, as noted in [109]. Like most approaches, IPC assumes that odometry measurements are noisy but not corrupted by outliers: only the consistency of loop closure constraints is evaluated incrementally. For each newly detected loop closure $\mathbf{z}_{a,b}$, IPC evaluates its consistency by first selecting the *minimal independent subgraph* (defined be-

low) that fully includes the candidate constraint plus all other previous constraints that involve some of the nodes included in the loop. The subproblem associated with this subgraph is then optimized and, if the obtained solution disagrees with the previous solution obtained, the candidate loop closure is rejected, otherwise, it is added to the set of loop closure inliers ${}^i\mathcal{E}_l$, and the subgraph optimization is propagated to the rest of the graph.

Formally, given a pose graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define a *subgraph* $\mathcal{G}(a, b) = (\mathcal{V}(a, b), \mathcal{E}(a, b))$, with $a < b$ and $b - a > 1$, where $\mathcal{V}(a, b) = \{\mathbf{x}_i \in \mathcal{V} \mid a \leq i \leq b\}$ and $\mathcal{E}(a, b) = \{\mathbf{e}_{i,j} \in \mathcal{E}_o \cup {}^i\mathcal{E}_l \mid a \leq i \leq j \leq b\}$. $\mathcal{G}(a, b)$ includes all nodes, and related edges*, from index a to index b , where $[a, b]$ is a discrete time interval. The subgraph $\mathcal{G}(a, b)$ is defined as *independent* if it does not exist a loop closure constraint in ${}^i\mathcal{E}_l \setminus \mathcal{E}_l(a, b)$ that connects a node $\mathbf{x}_k \in \mathcal{V} \setminus \mathcal{V}(a, b)$ to a node $\mathbf{x}_i \in \mathcal{V}(a, b)$. If this independence condition is satisfied, then $\mathcal{G}(a, b) = \mathcal{G}^I(a, b) = \mathcal{G}^I$, and the subgraph, representing a subproblem, can be optimized independently. Whenever a candidate loop closure $\mathbf{z}_{a,b}$ is detected, IPC searches for the *minimal* independent subgraph $\mathcal{G}^I(a, b)$ that includes the new edge (a, b) . Fig. 5.2 reports an example with an edge (the red edge (7, 12)) representing a candidate loop closure, and the associated minimal independent subgraph (highlighted by the purple dotted rectangle).

IPC optimizes the following problem:

$$\sum_{(a,a+1) \in \mathcal{E}_o^I} \mathbf{e}_{a,a+1}^T s \Omega_{a,a+1} \mathbf{e}_{a,a+1} + \sum_{(a,b) \in \mathcal{E}_l^I} \mathbf{e}_{a,b}^T \Omega_{a,b} \mathbf{e}_{a,b} \quad (5.4)$$

The above equation differentiates from Eq. (5.3) as it uses a different set of measurements for the optimization process, and introduces the fixed parameter s which assigns more weight or *confidence* to do the odometry constraints during the optimization. This choice is based on the fact that odometry constraints are treated as inliers when solving such problems as noted in [109]. After the optimization of the associated subproblem, to check whether the new measurement agrees with the other already accepted constraints, authors exploit the χ^2 test [101]:

$$\mathbf{e}_{a,b}^T \Omega_{a,b} \mathbf{e}_{a,b} < \chi_{\alpha, \delta}^2, \quad \forall \mathbf{e}_{a,b} \in \mathcal{E}_o^I \cup {}^i\mathcal{E}_l \quad (5.5)$$

where α is the confidence parameter and δ the degrees of freedom of the error and $\mathcal{E}_o^I \cup {}^i\mathcal{E}_l$ is the set of edges of $\mathcal{G}^I$ associated with inlier measurements. If one of the tests of Eq. (5.5) fails, the new loop closure is classified as an outlier, and the corresponding edge is added to the outlier set ${}^o\mathcal{E}_l$. If all the tests give a positive result, the new constraint is accepted and included in the inlier set ${}^i\mathcal{E}_l$. The result of the optimization of $\mathcal{G}^I$ can hence be propagated to all other nodes $\mathbf{x}_i \in \mathcal{G} \setminus \mathcal{G}^I$ by simply applying the estimated transformation updates from the nodes in $\mathcal{G}^I$ to which they are connected.

To summarize, for each candidate loop closure $\mathbf{z}_{a,b}$ to be evaluated, IPC performs the following steps:

*Note that only inlier loop closure measurements are taken into account.

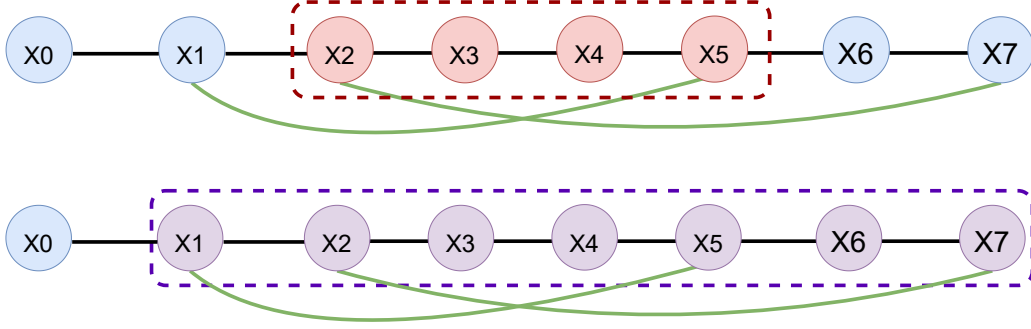


Figure 5.3: An example displaying the resulting intersection and union between the edges (x_1, x_5) and (x_2, x_5) . The loop closures are displayed in green, while the odometry edges are in black. In the top image, the red-dotted rectangle highlights which variables belong to the *intersection* of the loop closures. In the bottom image, the purple-dotted rectangle highlights which variables belong to the *union* of the loop closures. The two sets have cardinality 4 and 7, respectively, leading to a $IoU = \frac{4}{7}$.

1. Search for the minimal independent subgraph $\mathcal{G}^I(a, b)$ including (a, b) ;
2. Optimize the associated subproblem using Eq. (5.4);
3. Test the new solution with Eq. (5.5) for each already accepted constraint.
4. If one of the tests fails, reject the candidate loop closure (${}^o\mathcal{E}_l = \{(a, b)\} \cup {}^o\mathcal{E}_l$), otherwise accept it (${}^i\mathcal{E}_l = \{(a, b)\} \cup {}^i\mathcal{E}_l$) and propagate the subgraph optimization to the rest of the graph.

Although we demonstrated that IPC can outperform other methods in handling large numbers of outliers, IPC suffers from two important shortcomings that limit its applicability from two points of view:

- **Scalability:** As the robot revisits the same locations, the minimal independent subgraph may grow until it converges to the original graph, significantly increasing the time required for a single consistency check;
- **Robustness:** As a greedy approach, IPC is unable to revise bad decisions eventually made in past iterations.

In the next sections, we present our contributions on how to overcome these limitations.

5.3.2 KBL-IPC: K-BEST LOOPS IPC

To address the scalability issues of IPC, we introduce an approximate version of the minimal independent subgraphs, which considers at most k *most influential* (“best”) loop closure detection constraints for the evaluated candidate. The idea is to maximize the overlap between

the candidate loop closure and the k loop closures taken into consideration while minimizing the cardinality of the nodes that belong to the selected subgraph. To achieve this, we propose using the Intersection over Union (IoU) metric to compare the sets of nodes defined by the constraints.

Let's consider a pair of loop closure constraints $\mathbf{e}_{a,b}$ and $\mathbf{e}_{c,d}$, with $a < b$ and $c < d$. They involve the two sequences of nodes, $\mathcal{V}(a, b) = \{\mathbf{x}_i \in \mathcal{V} \mid a \leq i \leq b\}$ and $\mathcal{V}(c, d) = \{\mathbf{x}_i \in \mathcal{V} \mid c \leq i \leq d\}$, respectively. The intersection of these sets is defined as $\mathcal{V}(a, b) \cap \mathcal{V}(c, d) = \{\mathbf{x}_i \in \mathcal{V}(a, b) \wedge \mathbf{x}_i \in \mathcal{V}(c, d)\}$, and the union is defined as $\mathcal{V}(a, b) \cup \mathcal{V}(c, d) = \{\mathbf{x}_i \in \mathcal{V}(a, b) \vee \mathbf{x}_i \in \mathcal{V}(c, d)\}$. We define the IoU between $\mathbf{e}_{a,b}$ and $\mathbf{e}_{c,d}$ as:

$$\text{IoU}(\mathbf{e}_{a,b}, \mathbf{e}_{c,d}) = \frac{|\mathcal{V}(a, b) \cap \mathcal{V}(c, d)|}{|\mathcal{V}(a, b) \cup \mathcal{V}(c, d)|} \quad (5.6)$$

where the operator $|\cdot|$ defines the cardinality of a node set (e.g., see Fig. 5.3).

For each incoming loop closure candidate $\mathbf{z}_{ij}$, instead of considering all the inlier set ${}^i\mathcal{E}_l$ to search for the minimal independent subgraph associated with this constraint as done in IPC, we select the k already accepted loop closure edges $\mathcal{E}_l^k \in {}^i\mathcal{E}_l$ that have the highest IoU with the edge $\mathbf{e}_{ij}$. The pseudo-code of the selection process is reported in Alg. 6. We then build the

Algorithm 6: findKBestLoops($\mathbf{e}_{a,b}, {}^i\mathcal{E}_l, k$)

```

1  $\mathcal{E}_l^k, \mathcal{E}^{tmp} \leftarrow \emptyset$ 
2 for (each edge  $\mathbf{e}_{i,j} \in {}^i\mathcal{E}_l$ ) do
3    $score \leftarrow \text{IoU}(\mathbf{e}_{a,b}, \mathbf{e}_{i,j})$ 
4    $\mathcal{E}^{tmp} \leftarrow \mathcal{E}^{tmp} \cup \{(score, \mathbf{e}_{i,j})\}$ 
   // Selecting K best loops
5  $\mathcal{E}^{tmp} \leftarrow \text{SORTBYDESCENDINGSCORE}(\mathcal{E}^{tmp})$ 
6 while ( $||\mathcal{E}_l^k|| < k$  and  $||\mathcal{E}^{tmp}|| > 0$ ) do
7    $\mathbf{e}_{i,j} \leftarrow \text{POPFRONT}(\mathcal{E}^{tmp})$ 
8    $\mathcal{E}_l^k \leftarrow \mathcal{E}_l^k \cup \{\mathbf{e}_{i,j}\}$ 
9 return  $\mathcal{E}_l^k$ 

```

minimal independent subgraph $\mathcal{G}^k(a, b) = \mathcal{G}^k$ associated with $\mathbf{e}_{ij}$ as described in Sec. 5.3.1 by considering the selected subset of loop closure edges $\mathcal{E}_l^k$, the candidate constraint $\mathbf{e}_{ij}$, and as usual the related edges associated with the odometry measurements. Fig. 5.2 reports an example with a candidate loop (the red edge (7, 12)), and a possible minimal independent subgraph $\mathcal{G}^k$ with $k = 2$ (highlighted by the blue dotted rectangle). The parameter k balances the trade-off between the speed and robustness of the proposed solution. Smaller k values reduce test time by preventing the degenerate case of reverting to the full problem $\mathcal{G}$. The consistency test is the same as the one defined in Eq. (5.5), but using $\mathcal{E}^k$ instead of $\mathcal{E}_l^I$. The same rules as IPC apply: if the new loop closure is deemed inconsistent, no further actions are required, as the system

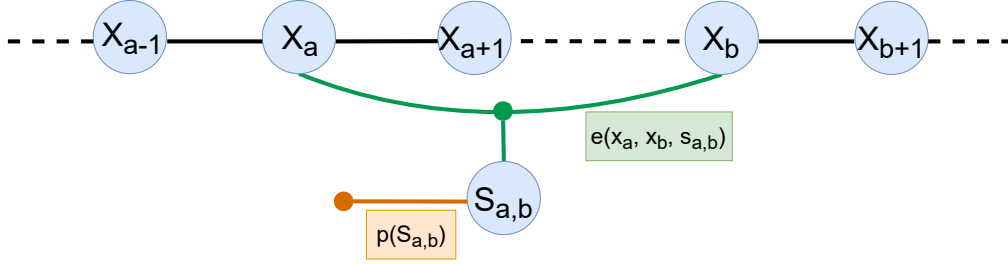


Figure 5.4: The image illustrates the switchable constraint formulation described in [108]. The green edge represents the ternary factor involving the variables $\mathbf{x}_a$, $\mathbf{x}_b$, and $s_{a,b}$, where $s_{a,b}$ serves as the switch variable, capable of activating or deactivating the constraint. The orange unary edge connected to $s_{a,b}$ represents the prior constraint on the loop closure edge.

reverts to its prior state. Conversely, if the new loop closure is accepted, the update must be propagated throughout the graph. This propagation is efficient, as only a limited number of new variables and constraints are typically integrated during the consistency check. We named this approximated version of IPC as kBL-IPC, which stands for k-Best Loops IPC.

5.3.3 SWITCHABLE CONSTRAINTS

As already pointed out, being greedy approaches, IPC and kBL-IPC may make bad decisions and include outliers into ${}^i\mathcal{E}_l$. This situation is more likely especially during the early stages of the incremental loop closure validation, when the inlier set ${}^i\mathcal{E}_l$ contains few or no elements. At this stage, a new candidate loop closure measurement may not intersect with any previously accepted loop closure. Consequently, the minimal independent subgraph used by IPC might include only odometry edges. The consistency check then relies solely on these measurements, which may not impose enough constraints to prevent the integration of outliers. In fact, least squares solvers aim to find a solution that satisfies all constraints, effectively distributing the error over all free edges. This could limit the magnitude of individual errors so that all tests of Eq. (5.5) are satisfied, possibly integrating an outlier. In IPC and kBL-IPC this decision is irreversible and could easily lead to optimization divergence. To overcome this robustness issue, we propose to integrate a method capable of retrospectively revising loop closure constraints added incrementally with kBL-IPC. Our approach builds upon the *SC* framework introduced in [108]. In *SC* the standard optimization problem for robust pose graph optimization has been enhanced adding the possibility to change the topology of the graph during optimization. This is achieved with the introduction of a new set of variables and constraints:

$$\hat{\mathbf{e}}_{a,b} = \mathbf{e}(\mathbf{x}_a, \mathbf{x}_b, s_{a,b}) = \Psi(s_{a,b})\mathbf{e}(\mathbf{x}_a, \mathbf{x}_b) \quad (5.7)$$

The Eq. (5.7) is an enhanced version of the loop closure error factor of Eq. (5.3). While the original loop closure factor depends solely on the variables $\mathbf{x}_a$ and $\mathbf{x}_b$, in Eq. (5.7) it is introduced an additional variable, $s_{a,b} \in \mathbb{R}$, referred to as the *switch* variable. This new variable controls

the weight of the loop closure constraints through the mapping function $\Psi(\cdot) : \mathbb{R} \rightarrow [0, 1]$, which remaps the switch variable into the $[0, 1]$ interval. Ideally, $\Psi(s_{a,b}) = 1$ when the associated loop closure $\mathbf{z}_{ab}$ is an inlier, and $\Psi(s_{a,b}) = 0$ when $\mathbf{z}_{ab}$ is an outlier. Setting $\Psi(s_{a,b}) = 0.0$ is the topological equivalent of disconnecting the variables $\mathbf{x}_a$ and $\mathbf{x}_b$ which were previously linked by the loop closure $\mathbf{z}_{ab}$. As reported in [108], better convergence can be obtained if we use a simple linear function $\Psi(s_{a,b}) = s_{a,b}$, as long as we constrain $0 \leq s_{a,b} \leq 1$. To prevent the optimizer from setting all loop closure weights to zero, an additional term is introduced in Eq. (5.3):

$$p_{a,b} = p(s_{a,b}) = \|\gamma_{a,b} - s_{a,b}\| \quad (5.8)$$

This penalization term relies on $\gamma_{a,b} \in \mathbb{R}$, which encodes prior knowledge on the inlier or outlier nature of the loop constraint $\mathbf{e}_{a,b}$. The full augmented problem is represented by the following cost function obtained by substituting the second error term in Eq. (5.3) with both Eq. (5.7) and Eq. (5.8):

$$\sum_{(a,a+1) \in \mathcal{E}_o} \mathbf{e}_{a,a+1}^T \Omega_{a,a+1} \mathbf{e}_{a,a+1} + \sum_{(a,b) \in \mathcal{E}_l} (\hat{\mathbf{e}}_{a,b}^T \Omega_{a,b} \hat{\mathbf{e}}_{a,b} + p_{a,b} \Omega_p p_{a,b}) \quad (5.9)$$

Please note that, since we want to revise the decisions made by the IPC/IPCKBL-IPC, unlike Eq. (5.3), in Eq. (5.9) we have considered the entire set of loop closure edges $\mathcal{E}_l$, regardless of whether they were classified as inliers or outliers.

5.3.4 SR-SC: SELECTIVE RANDOMIZED SWITCHABLE CONSTRAINTS

Direct integration of SC is not feasible due to its intrinsic weaknesses. First, SC increases the number of variables to be estimated in a highly non-convex problem like SLAM. This added complexity results in a more irregular convergence basin[114], making it harder for the optimizer to reach a favorable local optimum. The second issue lies in the default prior value $\gamma_{a,b}$ of the variable $s_{a,b}$, which is set to 1. This biases the optimizer toward accepting as many loop closures as possible by minimizing this error as well. In environments with a high percentage of outliers—such as caves or underwater, where aliasing is prevalent—this strategy for setting priors becomes suboptimal. To overcome these limitations, we propose to integrate the basic SC framework within a randomized voting scheme that acts on a subset of the original problem. We call this approach *Selective Random Switchable Constraints (SR-SC)* as it leverages a subset of constraints selected from previous steps while the voting scheme is performed over a series of random perturbations of the initial hypotheses. SR-SC will be periodically run on a set of R loop closures, possibly in parallel, to provide a retrospective revision of the decision made online by kBL-IPC.

In our method, we use kBL-IPC to incrementally populate the inlier set $\mathcal{E}_l$. Let $\bar{\mathcal{E}}_l$ be an auxiliary edge set we call *unrevised set* (see Fig. 5.1) that includes all the loop closure tested by kBL-IPC *but not yet revised with SR-SC*. $\bar{\mathcal{E}}_l$ is populated with both loop closure edges classified

by kBL-IPC as inliers and those classified as outliers. For each newly detected loop closure, after evaluating its consistency with kBL-IPC and populating either ${}^i\mathcal{E}_l$ or ${}^o\mathcal{E}_l$, the constraint is also added to $\bar{\mathcal{E}}_l$. This process continues until the cardinality of $\bar{\mathcal{E}}_l$ reaches a specified size R . At that point, we execute the SR-SC loop closure revision procedure that tests in batch all the edges included in $\bar{\mathcal{E}}_l$. This procedure can potentially revise the inlier-outlier classification previously made by kBL-IPC for $\bar{\mathcal{E}}_l$. To do this, we also leverage all the inlier loop closures already tested and revised in previous executions of SR-SC. Such constraints belong to the set ${}^t\mathcal{E}_l = {}^i\mathcal{E}_l \setminus \bar{\mathcal{E}}_l$ that we call *trusted set*. These inliers will be trusted during the revision procedure, meaning their inlier status will remain unchanged. Given these definitions, the first idea behind SR-SC is to optimize the following SC cost function derived from Eq. (5.9):

$$\sum_{(a,a+1) \in \mathcal{E}_o} \mathbf{e}_{a,a+1}^T \Omega_{a,a+1} \mathbf{e}_{a,a+1} + \sum_{(a,b) \in {}^t\mathcal{E}_l} \mathbf{e}_{a,b}^T \Omega_{a,b} \mathbf{e}_{a,b} + \sum_{(a,b) \in \bar{\mathcal{E}}_l} (\hat{\mathbf{e}}_{a,b}^T \Omega_{a,b} \hat{\mathbf{e}}_{a,b} + p_{a,b} \Omega_p p_{a,b}) \quad (5.10)$$

where the loop closures that belong to the trusted set ${}^t\mathcal{E}_l$, as well as the odometry measurements in $\mathcal{E}_o$, are considered inliers, while the loop closures that belong to the unrevised set $\bar{\mathcal{E}}_l$ are augmented with switch variables to enable inlier/outlier revision. Another basic idea in SR-SC is to initialize the switch variables priors $\gamma_{a,b} \in \mathbb{R}$ (see Eq. (5.8)) for the constraints in $\bar{\mathcal{E}}_l$ with the labels previously assigned by kBL-IPC, where $\gamma_{a,b} = 1$ indicates that the associate loop closure $\mathbf{e}_{ab}$ is an inlier and $\gamma_{a,b} = 0$ that it is an outlier. Although promising, these ideas have two flaws: Eq. (5.10) requires optimizing the entire graph, and in the presence of incorrect inlier/outlier labels, the optimization could get stuck in some local minima. To address this, SR-SC introduces two further strategies:

- The optimization of Eq. (5.10) is performed only on a subproblem associated with a subgraph $\mathcal{G}^T \in \mathcal{G}$ we call *minimal trusted subgraph* (see definition below) that includes all edges in $\bar{\mathcal{E}}_l$ and a minimal subset of edges that belong to the trusted set ${}^t\mathcal{E}_l$ and to the odometry set $\mathcal{E}_o$;
- The optimization is repeated with a perturbation of the switch priors until convergence is achieved;
- The new set of inliers in the unrevised set $\bar{\mathcal{E}}_l$ is selected by a voting scheme that counts how many times they were selected as inliers in the previous step.

We define $\mathcal{G}^T = (\mathcal{V}^T, \mathcal{E}^T) \in \mathcal{G}$ to be a trusted subgraph for an unrevised set $\bar{\mathcal{E}}_l$ if:

- It includes all the edges in the unrevised set, i.e. $\bar{\mathcal{E}}_l \in \mathcal{E}^T$
- It includes all the vertices connected by the edges in $\bar{\mathcal{E}}_l$, i.e.:

$$\bar{\mathcal{V}}_l = \{\mathbf{x}_a, \mathbf{x}_b \in \mathcal{V} \mid \mathbf{e}_{ab} \in \bar{\mathcal{E}}_l\} \in \mathcal{V}^T \quad (5.11)$$

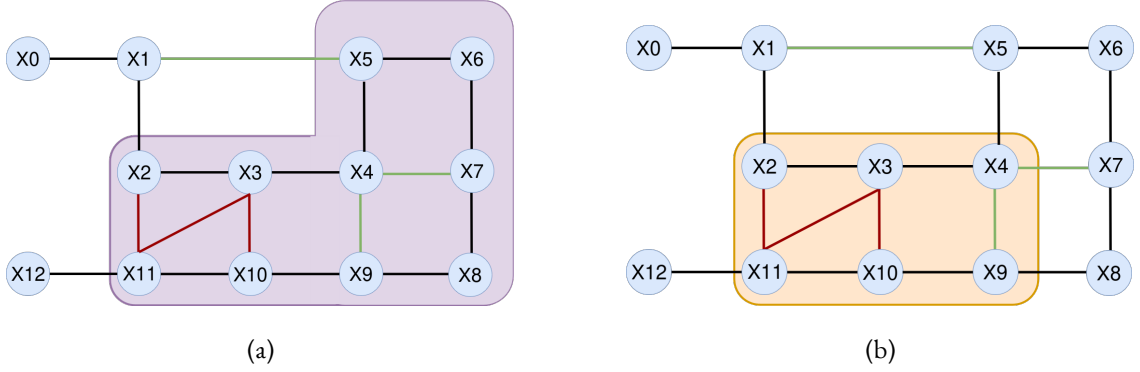


Figure 5.5: An example of two possible trusted subgraphs used for SR-SC for the same unrevised set $\bar{\mathcal{E}}_l = \{(\mathbf{x}_2, \mathbf{x}_{11}), (\mathbf{x}_3, \mathbf{x}_{11}), (\mathbf{x}_3, \mathbf{x}_{10})\}$ (marked in red). Loops belonging to the trusted set ${}^t\mathcal{E}_l$ are marked in green, while odometry edges are shown in black. In (a), a general trusted subgraph is highlighted by the purple shape, while in (b), the corresponding minimal trusted subgraph is highlighted by the orange rectangle.

- For each pair of vertices $\mathbf{x}_a, \mathbf{x}_b \in \bar{\mathcal{V}}_l$, there exists a path composed only by trusted and odometry edges, i.e. edges either in ${}^t\mathcal{E}_l$ or in $\mathcal{E}_o$.

A minimal trusted subgraph for $\bar{\mathcal{E}}_l$ is the trusted subgraph with the minimum quantity of vertices and edges for $\bar{\mathcal{E}}_l$. To find the minimal subgraph under this setting, the Dijkstra algorithm [121] is used as it finds the shortest path to every other node in the graph. In Fig. 5.5 two examples of trusted subgraphs for the same set of edges (i.e., the red edges). The subgraph highlighted by the orange rectangle is minimal. Given a minimal trusted subgraph $\mathcal{G}^T = (\mathcal{V}^T, \mathcal{E}^T)$ for $\bar{\mathcal{E}}_l$, with corresponding trusted set ${}^t\mathcal{E}_l = {}^i\mathcal{E}_l \setminus \bar{\mathcal{E}}_l$, the cost function of Eq. (5.10) can be reduced as:

$$\sum_{(a,a+1) \in \mathcal{E}_o \cap \mathcal{E}^T} \mathbf{e}_{a,a+1}^T \Omega_{a,a+1} \mathbf{e}_{a,a+1} + \sum_{(a,b) \in {}^t\mathcal{E}_l \cap \mathcal{E}^T} \mathbf{e}_{a,b}^T \Omega_{a,b} \mathbf{e}_{a,b} + \sum_{(a,b) \in \bar{\mathcal{E}}_l} (\hat{\mathbf{e}}_{a,b}^T \Omega_{a,b} \hat{\mathbf{e}}_{a,b} + p_{a,b} \Omega_p p_{a,b}) \quad (5.12)$$

The resulting problem is simpler to solve, increasing the likelihood of finding a good solution [114][20], while also requiring less time. The full SR-SC algorithm is reported in Alg. 7, where are also outlined the iterative optimization with random initialization and the final voting scheme.

After selecting the trusted set and computing the minimal trusted subgraph (lines 1,2), we perform an initial optimization of the parameters of $\mathcal{G}^T$ without using the unrevised set $\mathcal{E}_l$ and the related switchable constraints (line 3). These parameters will be used as initial guesses in each subsequent optimization. For each iteration of the loop in line 5, we initialize the switch variables priors $\gamma_{a,b}$ with the labels previously assigned by kBL-IPC, with a small percentage ϵ of flipping this estimate (lines 6, 7). As an additional implementation detail (not reported in

Algorithm 7: SR-SC($\mathcal{G}, {}^i\mathcal{E}_l, \bar{\mathcal{E}}_l$)

```
1  ${}^t\mathcal{E}_l = {}^i\mathcal{E}_l \setminus \bar{\mathcal{E}}_l$ 
2  $\mathcal{G}^T \leftarrow \text{FINDMINTRUSTEDGRAPH}(\mathcal{G}, {}^i\mathcal{E}_l, \bar{\mathcal{E}}_l)$ 
   // Find an initial guess for  $\mathcal{G}^T$  using
   // only trusted loop closures and odometry
3 solve Eq. (5.12) without considering constraints in  $\bar{\mathcal{E}}_l$ 
   // Initialize the accumulator
   // used in the voting scheme
4  $\forall (a, b) \in \bar{\mathcal{E}}_l : acc_{a,b} \leftarrow 0$ 
5 for (maxIter times or until convergence) do
   // Initialize switch variables priors
   // with kBL-IPC
6  $\forall (a, b) \in \bar{\mathcal{E}}_l : \gamma_{a,b} \leftarrow 1$  if  $(a, b) \in {}^i\mathcal{E}_l$ , 0 otherwise
   // Perturbate a few priors
7  $\forall (a, b) \in \bar{\mathcal{E}}_l : \gamma_{a,b} \leftarrow \text{RANDOMFLIP}(\epsilon)$ 
8 solve Eq. (5.12)
   // Check for inliers
   // and update the accumulator
9 for (each  $(a, b) \in \bar{\mathcal{E}}_l$ ) do
10   if  $s_{a,b} > 0.5$  then
11      $acc_{a,b} \leftarrow acc_{a,b} + 1$ 
12  $\mathcal{E}^{tmp} \leftarrow \emptyset$ 
13 for (each  $(a, b) \in \bar{\mathcal{E}}_l$ ) do
14    $avgS \leftarrow \text{GETAVERAGESTATE}(acc_{ab})$ 
15   if  $avgS > inlierTh$  then
16      $\mathcal{E}^{tmp} \leftarrow \mathcal{E}^{tmp} \cup \{(a, b)\}$ 
   // Update the inlier set
17  ${}^i\mathcal{E}_l = {}^t\mathcal{E}_l \cup \mathcal{E}^{tmp}$ 
   // Clear the unrevised set
18  $\bar{\mathcal{E}}_l \leftarrow \emptyset$ 
```

Alg. 7), some of the switchable variables $s_{a,b}$ are also randomly fixed, intentionally biasing the optimization to preserve those estimates. After solving the full problem of Eq. (5.12) with the new initial guess, the switch variables are evaluated, and the corresponding accumulator bins are incremented (lines 9-11). Intuitively, loop closures selected as inliers multiple times regardless of the initial guess are more likely to be validated. The outer loop (lines 5-11) stops upon convergence, which means that even when changing the initial configuration of the subproblem, the estimates of the switch variables remain unchanged w.r.t. the previous iteration, or after *maxIter* iterations. Loop closure detections frequently classified as inliers are promoted

to inliers by SR-SC (lines 13-16). Finally, the unrevised set $\bar{\mathcal{E}}_l$ is cleared, ready to be filled again during subsequent kBL-IPC iterations (line 18).

5.4 EXPERIMENTS

The proposed method was implemented in three variations, named:

- kBL-TACO, that is kBL-IPC + SR-SC;
- TACO, that is IPC + SR-SC;
- kBL-IPC alone.

The implementation was developed in C++, leveraging the g2o framework [86], along with the IPC and Switchable Constraints [108] implementations. Dog-Leg optimization steps were utilized, with s set to 10 and 30 for the 2D and 3D Visual SLAM experiments, respectively. SR-SC is executed every 10 loop closure detections which is equivalent to the maximum size $R = 10$ allowed to the unrevised set $\bar{\mathcal{E}}_l$. In Alg. 7, the maximum number of SR-SC iterations $maxIter$ is limited to 20, the percentage ϵ of randomly flipped inlier/outlier labels is set to 0.3 while $inlierTh$ is set to 0.7. We compare the variants of the proposed method with five recent state-of-the-art approaches for RPGO, detailed below along with their acronyms used in the tables and plots and the corresponding implementations. For Max-Mixture method [104] (MAXMIX in the tables) in the experiment we used the implementation provided in OpenSLAM[†]. For Pairwise consistent measurement set maximization method [99] (PCM in the tables), Kimera’s implementation is used [115]. The remaining methods are provided by the gtsam library [85]: Graduated Non-Convexity [5], Dynamic Covariance Scaling [102], and Huber [103], will be referred to in the tables and plots respectively as GNC, DCS, and HUBER. For all methods we kept all their parameters constant across different datasets with same dimensionality (2D and Visual SLAM have different parameters). Experiments were carried out in a cluster PC using a single Intel(R) Xeon(R) Gold 5220 CPU. As done in [106], for methods that don’t explicitly classify measurements, an inlier is defined as a measurement having a χ^2 error less than χ_{th}^2 with $\alpha_{th} = 0.95$. For all methods, the different (loop closure) inliers sets are evaluated at the end of the optimization of the full problem, and for the evaluation of the SR, $th_{ATE} = 0.75$.

5.4.1 METRICS

In SLAM experiments the estimated trajectory $\mathbf{x}^*$ is usually compared against the ground truth $\mathbf{x}^{gt}$ using two metrics: **Absolute Trajectory Error (ATE)** and **RPE (RPE)** [120]. To compute the ATE, the first step is to find the rigid-body transformation $\mathbf{x}^{ATE} \in \text{SE}(3)$ that maximally

[†]<https://openslam-org.github.io/>

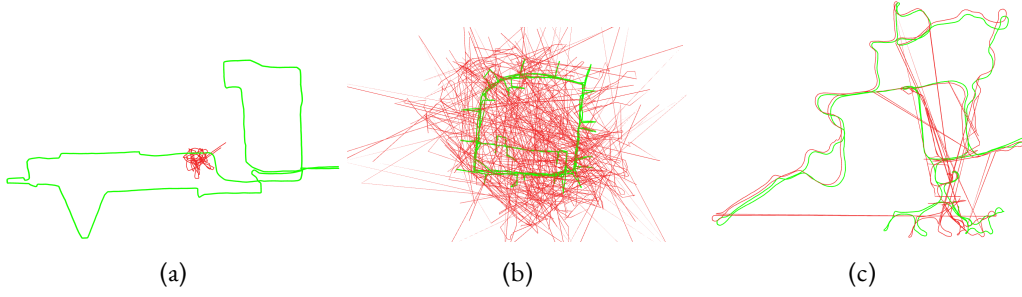


Figure 5.6: Image representing three distortion effects that arise from incorrectly integrating outliers into the optimization process.

aligns the two trajectories $\mathbf{x}^*$ and $\mathbf{x}^{gt}$. This transformation is then applied to the estimated trajectory and the position errors between corresponding nodes are computed.

The RPE, on the other hand, focuses on local accuracy by comparing the relative transformations between consecutive poses in the estimated and ground-truth trajectories. Both ATE and RPE are computed as the Root Mean Square Error (RMSE) over their respective errors.

These two metrics effectively evaluate SLAM system accuracy, with ATE measuring global trajectory accuracy and RPE focusing on local accuracy. However, they are not designed to assess robustness in datasets with significant outliers. The results are highly influenced by the type of distortion introduced by the outliers (e.g., see Fig. 5.6), making it difficult to select and properly configure a robust solver.

Fig. 5.7 illustrates various distorted trajectories along with their ATE, RPEs, and corresponding ground truth. Trajectories in (a1) and (b1) can be viewed as being composed of smaller trajectory segments. In (a2) each segment is moved away from each other, but locally they retained their shape. (b2) shows only one segment being distorted, with the rest of the trajectory remaining nearly unaltered. In (a3) and (b3), the distortion is so severe that the trajectory collapses on itself as the optimizer attempts to minimize the various errors. Qualitatively, it can be appreciated that the solutions of (a2) and (b2) are superior to those in (a3) and (b3). Yet, the ATE and RPE scores suggest the opposite.

Commonly, the average of the ATE and RPE are used as metrics to define which robust optimizer to choose when tackling the problem of RPGO. Unfortunately, the trajectories with high distortions can move the average of the ATE and RPE in unpredictable ways, possibly falsifying which solution is best. Failures and partial failures in PGO can result in different types of distortion of the real trajectory. In Fig. 5.6, the primary encountered distortions are depicted, and their effects on the ATE and RPE metrics are discussed below. In case (a), the trajectory collapses on itself, nearly degenerating into a point, as similarly seen in Fig. 5.7a3. The optimal ATE alignment, $\mathbf{x}^{ATE}$, moves the estimated trajectory $\mathbf{x}^*$ to the center of $\mathbf{x}^{gt}$, bounding the ATE to the average distance of the ground-truth poses $\mathbf{x}_i^{gt}$ from the trajectory's center, and the RPE to the average displacement between consecutive ground-truth poses $\mathbf{x}_i^{gt}$ and $\mathbf{x}_{i+1}^{gt}$. In case (b), severe disagreement between edges causes the trajectory to explode out-

5.4.2 METRICS FOR RPGO

As previously mentioned, ATE and RPE are meaningful metrics only when the algorithm converges to a solution that is not highly distorted. Instead of relying solely on ATE as an evaluation metric, we propose using the **Success Rate** (SR). The Success Rate (SR) is then defined as follows:

$$SR = \frac{N_{suc}}{N_{tot}}$$

where N_{suc} is the number of successful runs, and N_{tot} is the total number of the runs. A run is deemed successful if the estimated trajectory achieves an ATE below a defined threshold th_{ATE} , while an ATE exceeding th_{ATE} indicates a distorted trajectory. Consequently, the SR reflects the likelihood of a robust optimizer converging to a non-distorted trajectory. This likelihood is more informative than ATE values, which can fluctuate significantly and fail to reliably represent optimization performance as shown in Fig. 5.7. In support of this metric, which gives a more coarse overview of how the algorithms are performing, to have more fine-grained information we use the **Precision**, **Recall**, and **F1 score**:

$$Prec = \frac{TP}{TP + FP}, Rec = \frac{TP}{TP + FN}, F1 = 2 * \frac{Prec * Rec}{Prec + Rec}$$

The precision relays information regarding how many outliers have been integrated. Instead, the recall relays information about how many inliers have been discarded from the system. The F1 score relates both metrics and gives a more global idea of which systems are performing better. Ideally, both precision and recall should be 1. However, low precision poses a greater risk than low recall, as it can result in a highly distorted trajectory. In contrast, low recall primarily sacrifices drift removal, which is less detrimental overall.

5.4.3 DATASETS

The datasets used for evaluation are divided into two categories: 2D and 3D Visual SLAM. The 2D datasets are described in [116][117] were already available online. The chosen datasets—Intel, MIT, CSAIL, Freiburg Building, and Freiburg University Hospital—were all generated from real-world data. For the Visual SLAM evaluation, real-world datasets had to be adapted to the RPGO task. Namely KITTI_00, KITTI_05, and TUM_FR1_DESK. KITTI_00 and KITTI_05 were derived from sequences 00 and 05 of KITTI’s visual odometry dataset [122], while TUM_FR1_DESK was extracted from the corresponding sequence in TUM’s RGB-D SLAM Dataset [123]. Odometry and loop closure constraints were obtained using ORB_SLAM2 on the raw data from these sequences. The loop closure edges in the unaltered datasets served as ground truth for inliers. The solution obtained using SE-Sync [119], which is a fast non-minimal solver for PGO, on the outlier free version of the dataset was used as ground truth trajectory for the evaluation of the ATE [120]. Outliers were artificially introduced using the Vertigo package [118] as a percentage of the total inliers to ensure balanced results. Each dataset

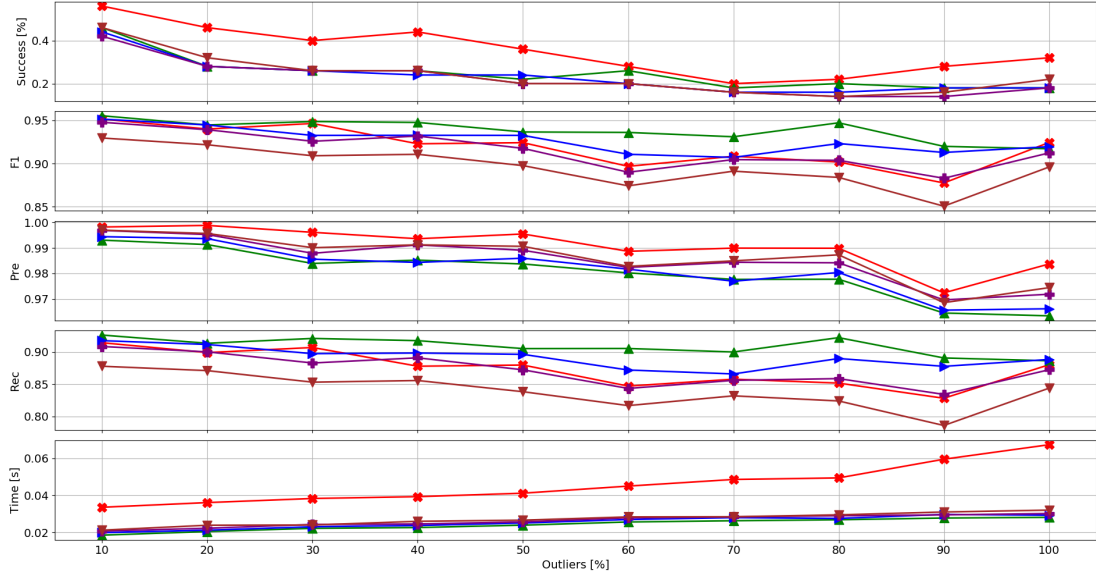


Figure 5.8: A comparison of the performance of kBL-IPC as the parameter k varies on 2D datasets. The metrics are presented from top to bottom: Success Rate, F1 Score, Precision, Recall, and Average Time per consistency check. Legend of the graphs: 2BL-IPC(▲), 3BL-IPC (►), 5BL-IPC(✚), 10BL-IPC(▼), and IPC(✕).

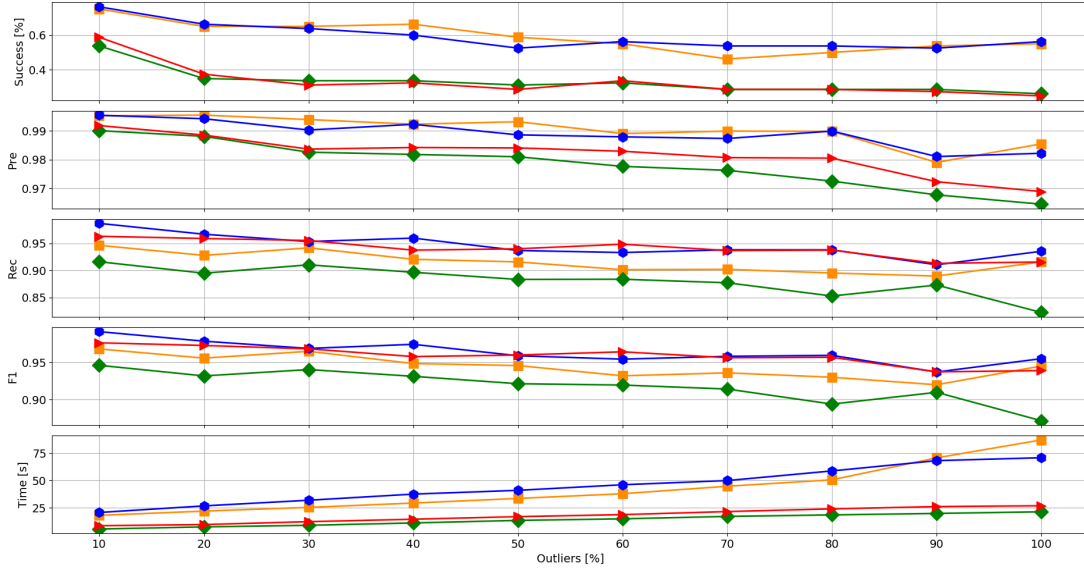


Figure 5.9: A comparison of the performance between IPC, TACO and 2BL-IPC on 2D and Visual SLAM datasets. The metrics are presented from top to bottom: Success Rate, F1 Score, Precision, Recall, and Completion Time. Legend of the graphs: TACO(●), 2BL-TACO(►), IPC(■), 2BL-IPC(◆).

includes ten difficulty levels based on the percentage of outliers, ranging from 10% to 100% of the ground truth inliers, in increments of 10%. At 100%, the number of outliers equals the number of inliers. To ensure robust and fair evaluation, ten distinct corrupted datasets were generated for each outlier percentage.

5.4.4 EFFECT OF THE PARAMETER k IN k BL-IPC

In this section, we evaluate the effect of the parameter k on the robustness of consistency tests and, consequently, the overall performance of k BL-IPC. When k is set to include all intersected loop closures, the algorithm reverts to the standard IPC. As expected, reducing the number of constraints involved in the consistency checks negatively impacts the SR Fig. 5.8. Among the variations, 2BL-IPC slightly outperforms 3BL-IPC, 5BL-IPC, and 10BL-IPC in terms of SR and Average Convergence Time per Constraint. The reduced SR relative to IPC is offset by a significant improvement in speed for 2BL-IPC. It is interesting to see that even if 2BL-IPC has a higher F1 score than IPC, its SR is still lower. This phenomenon can be explained by Fig. 5.8, which shows that IPC has the highest precision. Even when the precision’s difference seems negligible, the integration of a few extra outliers can significantly affect the optimization’s success and the overall system’s SR. IPC’s conservative approach during consistency checks—reflected in its slightly lower Recall—enables it to achieve better overall performance. In general, 2BL-IPC is, in average, 1.73 times faster than IPC, gained 1.9% in F1 score, and lost 10.4% in SR.

5.4.5 IPC vs IPC+SR-SC

In this section, we analyze the effects of adding SR-SC on top of IPC and 2BL-IPC. TACO and 2BL-TACO will refer, respectively, to the variations of IPC and 2BL-IPC with the additional recovery procedure. As shown in Fig. 5.9, the SR of TACO slightly outperforms IPC. On average, it gains 1.24% on the SR thanks to the 3.03% added percentage points on the Re-

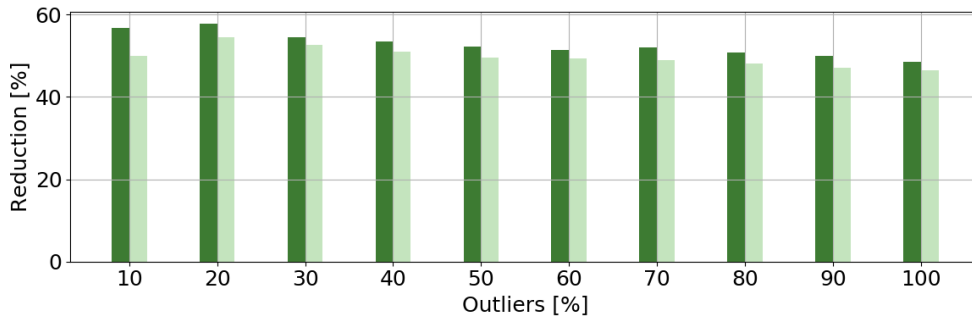


Figure 5.10: The average reduction in problem size across all datasets, both 2D and Visual SLAM, is shown for varying outlier percentages, comparing TACO (dark green) and 2BL-TACO (light green).

call, while keeping almost equivalent scores on the Precision. The overhead added by SR-SC is only 1.14% of the time it would have taken IPC without recovery. For 2BL-IPC and 2BL-TACO, the SR stays the same, even if 2BL-TACO shows 4.1% and 5.96% improvement in both F1 score and Recall, while maintaining a slight improvement in Precision of 0.47% in average. Here the SR-SC overhead corresponds to only a 1.31% increase in the total time. This was made possible by reducing the problem’s size when applying recovery, as the size is one factor that may impact the complexity of the problem, and thus the convergence speed. The reduction strategy utilized by TACO and 2BL-TACO (see description of minimal trusted subgraph in Sec. 5.3.4) consistently reduced the problem size to approximately half of its original, as illustrated in Fig. 5.10.

5.4.6 COMPARISON WITH STATE-OF-THE-ART METHODS

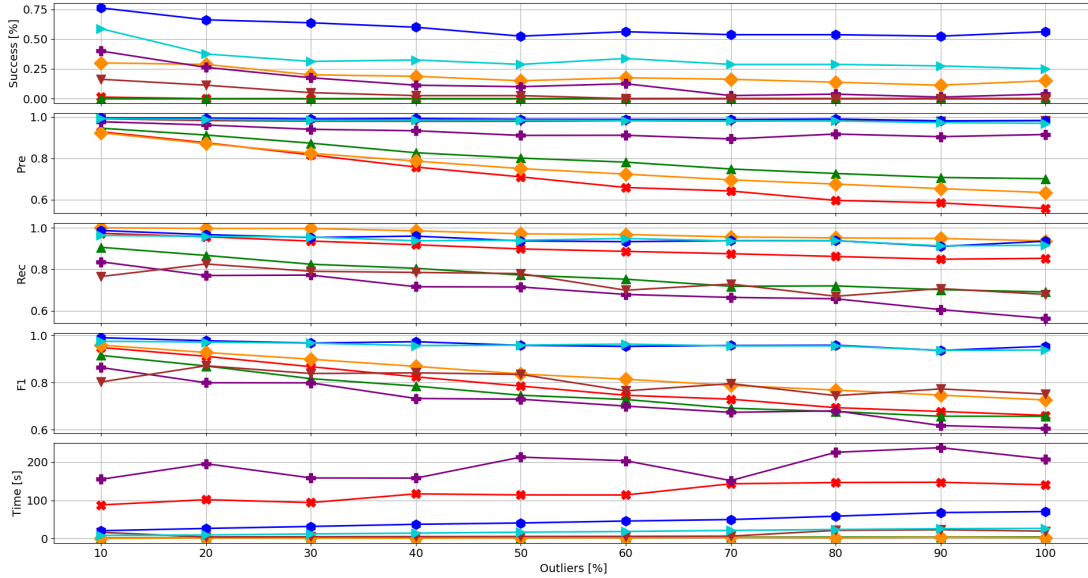


Figure 5.11: A comparison of the performance between state-of-the-art methods, TACO, and 2BL-TACO on 2D and Visual SLAM datasets as the outlier percentage varies. The metrics are presented in the following order from top to bottom: Success Rate, F1 Score, Precision, Recall, and Completion Time. Legend of the graphs: HUBER ($\blacktriangle$), TACO ($\bullet$), 2BL-TACO ($\blacktriangleright$), DCS ($\blacklozenge$), MAXMIX ($\blacktriangledown$), PCM ($\oplus$), GNC ($\times$).

Fig. 5.11 illustrates the performance metrics—success rate, precision, recall, F1 score, and execution time—averaged across all the datasets used, both 2D and Visual SLAM. The plots are analyzed sequentially from top to bottom. The SR reveals a marked difference between the existing methods and TACO or 2BL-TACO. TACO achieves a valid solution over 55% of the time, while 2BL-TACO, its approximated version, does so more than 35% of the time. In contrast, all other methods fail to exceed a 20% SR. The closest one to achieve at least a 20%

Method	20%			40%			60%			80%			100%		
	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑
HUBER	0.0	0.84	0.94	0.0	0.74	0.86	0.0	0.67	0.82	0.0	0.61	0.75	0.0	0.58	0.72
DCS	0.24	0.90	0.83	0.2	0.83	0.71	0.22	0.77	0.62	0.2	0.71	0.55	0.22	0.67	0.50
PCM	0.14	0.69	0.95	0.08	0.59	0.92	0.1	0.54	0.89	0.02	0.50	0.90	0.02	0.40	0.91
GNC	0.0	0.90	0.89	0.0	0.80	0.76	0.0	0.71	0.65	0.0	0.66	0.59	0.0	0.63	0.56
MAXMIX	0.08	0.82	1.00	0.04	0.77	1.00	0.0	0.69	0.99	0.0	0.64	0.99	0.0	0.64	0.99
IPC	0.5	0.94	1.00	0.5	0.92	0.99	0.32	0.90	0.99	0.3	0.90	0.99	0.32	0.92	0.98
2BL-IPC	0.3	0.94	0.99	0.32	0.95	0.99	0.28	0.94	0.98	0.22	0.95	0.98	0.22	0.92	0.96
2BL-TACO	0.32	0.97	0.99	0.32	0.97	0.99	0.3	0.97	0.98	0.22	0.96	0.98	0.22	0.94	0.97
TACO	0.52	0.97	1.00	0.44	0.97	0.99	0.36	0.93	0.99	0.36	0.94	0.99	0.38	0.93	0.98

Table 5.1: Table summarizing the performance of the methods on 2D datasets with varying percentages of injected outliers.

Method	20%			40%			60%			80%			100%		
	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑	SR ↑	F1 ↑	PR ↑
HUBER	0.10	0.92	0.86	0.03	0.86	0.77	0.00	0.82	0.72	0.00	0.79	0.69	0.00	0.78	0.68
DCS	0.57	0.97	0.95	0.23	0.94	0.92	0.13	0.90	0.90	0.07	0.86	0.88	0.07	0.83	0.86
PCM	0.47	0.99	0.97	0.27	0.98	0.96	0.33	0.96	0.95	0.27	0.97	0.94	0.13	0.93	0.92
GNC	0.13	0.92	0.86	0.00	0.86	0.76	0.0	0.80	0.67	0.00	0.76	0.61	0.0	0.71	0.56
MAXMIX	0.27	0.96	0.95	0.27	0.95	0.95	0.13	0.89	0.96	0.27	0.92	0.97	0.17	0.94	0.96
IPC	0.90	0.98	0.99	0.93	0.99	0.99	0.93	0.99	0.99	0.87	0.98	0.99	0.93	0.98	0.99
2BL-IPC	0.47	0.91	0.98	0.43	0.90	0.98	0.43	0.89	0.97	0.40	0.81	0.96	0.33	0.80	0.97
2BL-TACO	0.50	0.97	0.98	0.40	0.94	0.98	0.43	0.96	0.98	0.40	0.95	0.98	0.30	0.93	0.97
TACO	0.90	0.99	0.99	0.92	0.99	0.99	0.93	0.99	0.99	0.89	0.98	0.99	0.90	0.99	0.99

Table 5.2: Table summarizing the performance of the methods on Visual SLAM datasets with varying percentages of injected outliers.

Method	50%		100%	
	ATE ↓	RPE ↓	ATE ↓	RPE ↓
HUBER	52.19	24.18	53.55	27.67
DCS	43.48	29.03	55.10	42.64
PCM	43.50	27.32	56.15	31.32
GNC	43.86	34.36	48.56	37.63
MAXMIX	41.31	0.35	43.95	0.47
IPC	12.49	0.03	22.85	0.03
2BL-IPC	21.27	0.08	29.06	0.10
2BL-TACO	15.49	0.11	26.70	0.11
TACO	16.97	0.06	21.47	0.04

Table 5.3: Table summarizing ATE and RPE of the methods on 2D datasets injected with 50% and 100% outliers, respectively.

Method	50%		100%	
	ATE ↓	RPE ↓	ATE ↓	RPE ↓
HUBER	119.10	2.56	121.39	2.68
DCS	59.61	1.93	61.39	1.72
PCM	80.68	1.94	83.45	2.42
GNC	118.93	10.95	120.92	11.27
MAXMIX	119.16	0.25	121.49	0.61
IPC	17.28	2.11	9.05	2.04
2BL-IPC	63.75	2.49	92.01	2.61
2BL-TACO	50.75	2.46	70.00	2.46
TACO	11.66	2.07	8.99	2.05

Table 5.4: Table summarizing ATE and RPE of the methods on Visual SLAM datasets injected with 50% and 100% outliers, respectively.

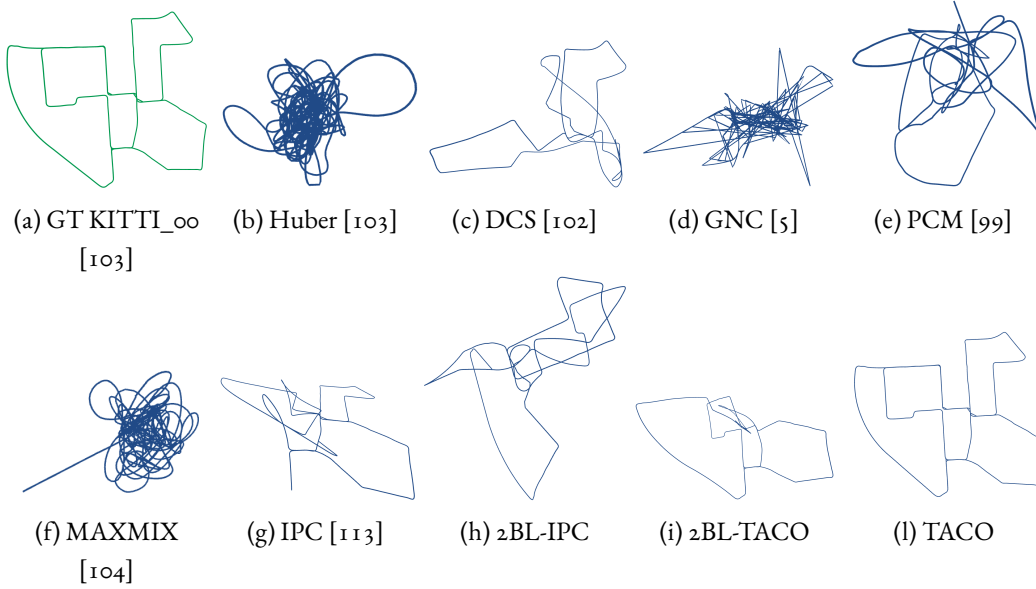


Figure 5.12: Qualitative analysis of the evaluated methods on the KITTI_00 trajectory with 100% outlier injection.

success rate is DCS.

Precision is one of the main metrics to consider when choosing a method: its value should ideally approach one since even minor outlier contributions have the potential to severely compromise the trajectory estimate. The precision plot from Fig. 5.11 provides insight into the degree of distortion a trajectory may experience, but it is not a 1 to 1 relation. Outliers can severely degrade a trajectory, meaning that not only the quantity but also the *quality* of integrated outliers plays a crucial role. The methods achieving the highest precision are TACO, 2BL-TACO, and MAXMIX. The precision performances of 2BL-TACO and MAXMIX are

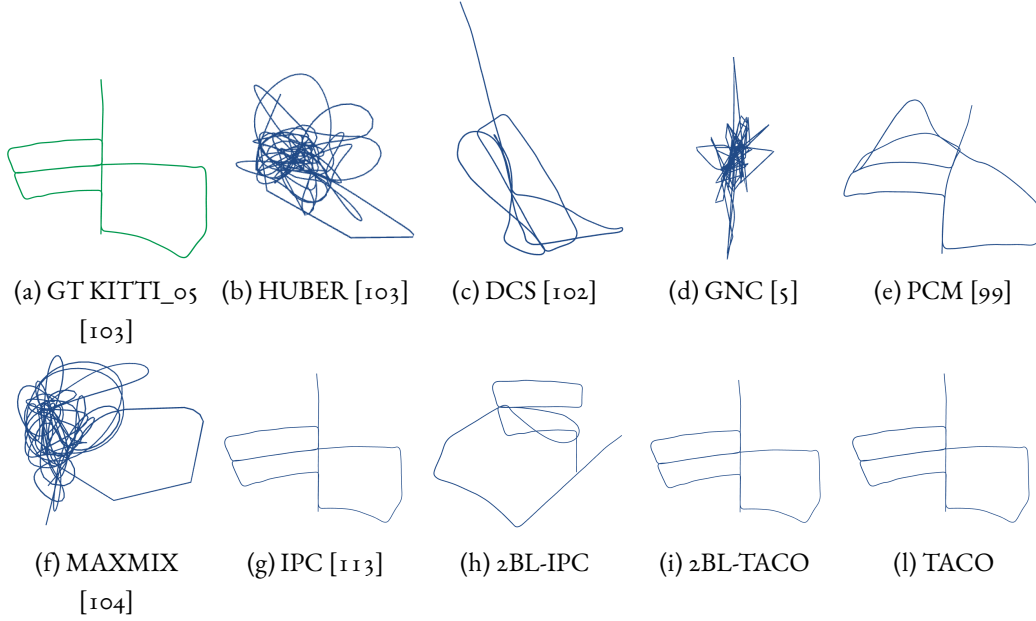


Figure 5.13: Qualitative analysis of the evaluated methods on the KITTI_05 trajectory with 70% outlier injection.

nearly identical, with MAXMIX’s trend being completely obscured by 2BL-TACO’s trend. Although HUBER, PCM, and MAXMIX demonstrate higher precision than DCS, the latter achieves a higher success rate than these methods. This discrepancy arises because failure is influenced not only by distortion but also by the inability to mitigate the accumulated drift. DCS also has the highest recall, followed by TACO, 2BL-TACO, and GNC. Even if GNC has a high recall, its poor precision prevents it from finding valid solutions. The F1 score gives a more global view of the method’s performance, as it relates precision and recall. TACO with IPC, closely followed by TACO with 2BL-IPC, are the methods with the highest F1 scores. These methods are able to maintain an F1 score, quasi-constant, always over 90%. That is to be expected, because to reach a valid solution both precision and recall must be close to 1. DCS is the next method that has the highest F1 score, its trend is very similar to its precision trend due to its high recall that it’s almost constant and close to 1.

We present the detailed results separately for the 2D and Visual SLAM datasets in Tab. 5.1 and Tab. 5.2, respectively. Qualitative trajectory reconstruction results for the KITTI_00 and KITTI_05 sequences are shown in Fig. 5.12 and Fig. 5.13. These figures illustrate the effectiveness of the TACO family of methods in preventing erroneous loop closures. For completeness, we also report in Tab. 5.3 and Tab. 5.4 the results for the ATE and RPE metrics for the 2D and Visual SLAM datasets, respectively. As introduced in Sec. 5.4.1, these metrics become uninformative when outliers are integrated into the optimization process, as the resulting trajectory is highly distorted. From the SR in Tables Tab. 5.1 and Tab. 5.2, we can observe that these distorted trajectories are a common occurrence when testing on datasets affected by a large

Method/Datasets	MIT	INTEL	CSAIL	FRH	FR079	FR1_DESK	KITTI_00	KITTI_05
HUBER	0.17	0.64	1.22	7.48	1.10	0.46	11.34	5.27
DCS	0.24	5.61	2.07	3.39	3.02	0.34	1.89	1.14
PCM	9.19	667.56	22.51	381.64	33.82	8.98	296.72	105.51
GNC	17.43	268.80	51.19	129.88	25.63	11.00	302.79	159.56
MAXMIX	2.91	43.72	2.52	23.78	1.96	0.58	11.09	4.89
IPC	0.67	20.14	6.25	255.84	8.57	1.12	33.59	9.17
2BL-IPC	0.49	18.22	5.56	45.28	5.19	0.83	28.44	6.57
2BL-TACO	1.27	19.59	6.68	64.08	7.25	1.12	33.39	9.97
TACO	1.47	20.39	8.20	253.32	10.51	1.33	52.92	12.99

Table 5.5: The runtime in seconds of the compared methods for each dataset, averaged across the different percentages of outliers.

number of outliers.

5.4.7 RUNTIME COMPARISON

Tab. 5.5 presents the runtimes of various methods for both 2D and 3D datasets, averaged across the various outlier percentages. As expected, HUBER and DCS are the fastest since M-Estimators only require reweighting the error function during optimization. In contrast, GNC and PCM exhibit longer execution times due to their additional computational demands: GNC solves multiple instances of the same problem, while PCM performs extensive compatibility testing. IPC and 2BL-IPC are purely incremental solutions, focusing solely on forward processing. In contrast, 2BL-TACO and TACO incorporate a retrospective check using SR-SC, which introduces additional overhead and increases runtime. The incremental nature of these methods ensures that their total runtime—defined as the time required to evaluate all loop closure edges in the dataset—scales linearly with the number of edges to test. Methods like 2BL-IPC and 2BL-TACO particularly reduce runtime in datasets such as FRH, which contains over 1400 inlier loops, in addition to the injected outliers, with respect to their counterparts IPC and TACO. Despite the overhead introduced by TACO, its total runtime remains comparable to that of IPC while delivering improved performance. In some cases, such as in the FRH dataset, TACO’s runtime is even shorter. This is attributed to the removal of outliers, which accelerates the convergence of the optimization process.

5.5 CONCLUSIONS

In this chapter, we introduced TACO, a robust optimization framework capable of incrementally optimizing a PGO while effectively rejecting most outliers through a combination of on-line tests and retrospective revisions. TACO outperforms recent state-of-the-art approaches and operates on two distinct time horizons: short-horizon and long-horizon. For the short

horizon, TACO utilizes a variant of IPC called kBL-IPC, which reduces the time required for IPC to make a decision by adjusting the parameter k , representing the number of constraints involved in the decision process. This results in faster execution, though it comes at a slight cost to the success rate. On the long horizon, SR-SC—a procedure based on switchable constraints inside a randomized voting scheme—was introduced to fix the mistakes made by kBL-IPC, albeit at a slower rate. Notably, TACO continues to outperform state-of-the-art methods, even with the faster kBL-IPC variation when $K=2$. The parameter k offers a flexible trade-off between speed and robustness, making it adaptable to various scenarios. TACO demonstrates strong generalization capabilities, delivering consistent results across diverse datasets without requiring parameter adjustments. However, the method’s current limitation lies in SR-SC’s reliance on kBL-IPC achieving reasonable performance, as its initial guess depends on kBL-IPC’s coherent set estimation. A potential future direction could involve identifying conditions under which kBL-IPC is guaranteed to perform effectively—based on factors such as problem size and constraint uncertainty—and developing a recovery procedure that remains as flexible but less dependent on kBL-IPC’s initial guess.

6

Conclusions

This thesis demonstrates the versatility of Pose Graphs as a powerful tool for addressing state estimation challenges, showcasing their ability to model, solve, and enhance the robustness of state estimation problems. By leveraging their structured probabilistic framework, we develop resilient solutions that improve both accuracy and reliability, ensuring dependable performance in complex and dynamic environments. This topic is of utmost importance as errors can significantly impact downstream tasks such as localization, mapping, and scene understanding, ultimately reducing the overall performance of the system.

Our first contribution demonstrates the effective use of PGs to model, solve, and enhance robustness in identification problems. Specifically, we introduce a PG-based formulation to address the Hand-Eye problem for a robotic arm operating within a workspace equipped with multiple cameras. This approach distinguishes itself from existing methods by seamlessly incorporating various camera models (e.g., pinhole, x-ray) and by introducing a novel *cross-observation* error term. This additional constraint links observations from different cameras capturing the calibration pattern at the same time step. The proposed pipeline proves to be highly resilient, maintaining calibration accuracy even when the input data is affected by visual and geometric errors. This application further underscores the flexibility of PGs, enabling robust calibration processes that mitigate the impact of noise and improve overall system reliability.

The second and third contributions of this thesis revolve around developing novel Robust Pose Graph Optimization frameworks designed to enhance the robustness of SLAM systems operating in highly ambiguous environments. Our second contribution introduces IPC, a Robust Pose Graph Optimization solution that handles outliers by evaluating measurement consistency through consensus. By incrementally evaluating each measurement, IPC builds an approximate solution to the inherently combinatorial problem of identifying the maximal set of coherent measurements.

Building upon IPC, our third contribution integrates the strengths of both offline and online methods into a single framework named TACO. TACO employs multi-horizon reason-

ing, where a short-horizon strategy enables rapid decision-making using an improved version of IPC, while a long-horizon approach provides robust error recovery through a voting scheme combined with a robust optimization framework. The proposed framework represents a further step towards achieving long-term autonomous navigation in highly ambiguous and unstructured environments.

To help advance the research on these topics, the implementations of our contributions are open source, so researchers can compare their new solutions with ours.

6.1 FUTURE WORK

Given the expressive capabilities of PGO for calibration problems, unifying various calibration methods within a common framework would offer significant practical benefits. Such a unified approach would streamline access to diverse calibration techniques, naturally enhancing their resilience to sensor noise. Additionally, incorporating covariance estimation into the calibration pipeline would further improve the methodology by mitigating the necessity to manually specify ad-hoc covariance matrices for each individual scenario [124].

Since PGO relies on iterative optimization, which depends on a good initial guess, an important and growing research direction is the development of methods for evaluating the quality of estimated results. The process of mathematically proving the reliability of these solutions, known as *certification* [12, 119], has gained significant attention in recent years. Certification will likely become a crucial aspect of future robotics, making it a research avenue worth exploring. In particular, a certifiable RPGO framework—rather than one reliant on heuristic validation—would be a game-changer for autonomous exploration in remote, highly aliased, and unstructured environments.

References

- [1] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, “The llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024.
- [2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Al-tenschmidt, S. Altman, S. Anadkat *et al.*, “GPT-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [3] J. Yu, C. Zhu, J. Zhang, Q. Huang, and D. Tao, “Spatial pyramid-enhanced netvlad with weighted triplet loss for place recognition,” *IEEE transactions on neural networks and learning systems*, vol. 31, no. 2, pp. 661–674, 2019.
- [4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, “Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, p. 1309–1332, 2016.
- [5] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, “Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1127–1134, 2020.
- [6] J. L. Schonberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.
- [7] L. Pan, D. Baráth, M. Pollefeys, and J. L. Schönberger, “Global structure-from-motion revisited,” in *European Conference on Computer Vision*. Springer, 2024, pp. 58–77.
- [8] A. Abdelkarim, H. Voos, and D. Gorges, “Factor graphs in optimization-based robotic control—a tutorial and review,” *IEEE Access*, 2025.
- [9] C. Hoffmann and P. Rostalski, “Linear optimal control on factor graphs—a message passing perspective—,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6314–6319, 2017.
- [10] H. Shabanian and M. Balasubramanian, “A novel factor graph-based optimization technique for stereo correspondence estimation,” *Scientific Reports*, vol. 12, no. 1, p. 15613, 2022.

- [11] J. Ou, P. Huang, J. Zhou, Y. Zhao, and L. Lin, “Automatic extrinsic calibration of 3d lidar and multi-cameras based on graph optimization,” *Sensors*, vol. 22, no. 6, p. 2221, 2022.
- [12] H. Yang, J. Shi, and L. Carlone, “Teaser: Fast and certifiable point cloud registration,” *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 314–333, 2020.
- [13] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, “Kiss-icp: In defense of point-to-point icp—simple, accurate, and robust registration if done the right way,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023.
- [14] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE transactions on robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.
- [15] L. Zhou, D. Koppel, and M. Kaess, “Lidar slam with plane adjustment for indoor environment,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7073–7080, 2021.
- [16] J. Dauwels, “On variational message passing on factor graphs,” in *2007 IEEE international symposium on information theory*. IEEE, 2007, pp. 2546–2550.
- [17] L. Schmid, J. Brenk, and L. Schmalen, “Local message passing on frustrated systems,” in *Uncertainty in Artificial Intelligence*. PMLR, 2023, pp. 1837–1846.
- [18] S. Thrun, “Simultaneous localization and mapping,” in *Robotics and cognitive approaches to spatial mapping*. Springer, 2008, pp. 13–41.
- [19] C. Stachniss, J. J. Leonard, and S. Thrun, “Simultaneous localization and mapping,” *Springer handbook of robotics*, pp. 1153–1176, 2016.
- [20] T. Guadagnino, L. Di Giammarino, and G. Grisetti, “Hipe: Hierarchical initialization for pose graphs,” *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 287–294, 2021.
- [21] T. Bailey, E. M. Nebot, J. Rosenblatt, and H. F. Durrant-Whyte, “Data association for mobile robot navigation: A graph theoretic approach,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3. IEEE, 2000, pp. 2512–2517.
- [22] V. Tzoumas, P. Antonante, and L. Carlone, “Outlier-robust spatial perception: Hardness, general-purpose algorithms, and guarantees,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5383–5390.

- [23] D. Zuñiga-Noël, J.-R. Ruiz-Sarmiento, R. Gomez-Ojeda, and J. Gonzalez-Jimenez, "Automatic multi-sensor extrinsic calibration for mobile robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2862–2869, 2019.
- [24] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1168–1174.
- [25] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4530–4537.
- [26] R. G. Golledge, "Place recognition and wayfinding: Making sense of space," *Geoforum*, vol. 23, no. 2, pp. 199–214, 1992.
- [27] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE transactions on robotics*, vol. 32, no. 1, pp. 1–19, 2015.
- [28] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph slam," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 1254–1261.
- [29] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2024.
- [30] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "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.
- [31] F. J. Radermacher, "Probabilistic reasoning in intelligent systems: Networks of plausible inference (judea pearl)," *SLAM Review*, vol. 32, no. 4, pp. 704–707, 1990.
- [32] J. R. Norris, *Markov chains*. Cambridge university press, 1998, no. 2.
- [33] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.
- [34] G. J. McLachlan, "Mahalanobis distance," *Resonance*, vol. 4, no. 6, pp. 20–26, 1999.
- [35] H. O. Hartley and A. Booker, "Nonlinear least squares estimation," *The Annals of mathematical statistics*, vol. 36, no. 2, pp. 638–650, 1965.

- [36] J. V. Burke and M. C. Ferris, “A gauss—newton method for convex composite optimization,” *Mathematical Programming*, vol. 71, no. 2, pp. 179–194, 1995.
- [37] J. J. Moré, “The levenberg-marquardt algorithm: implementation and theory,” in *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*. Springer, 2006, pp. 105–116.
- [38] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.
- [39] P. Rosenbloom, “The method of steepest descent,” in *Proc Symp Appl Math*, vol. 6, 1956, pp. 127–176.
- [40] T. B. Schon and F. Lindsten, “Manipulating the multivariate gaussian density,” *Div. Automat. Control, Linköping Univ., Linköping, Sweden, Tech. Rep.*, vol. 4, no. 3.4, p. 4, 2011.
- [41] J.-Y. Yuan and A. N. Iusem, “Preconditioned conjugate gradient method for generalized least squares problems,” *Journal of computational and applied mathematics*, vol. 71, no. 2, pp. 287–297, 1996.
- [42] T. A. Davis, *Direct methods for sparse linear systems*. SIAM, 2006.
- [43] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.
- [44] R. Wolke and H. Schwetlick, “Iteratively reweighted least squares: algorithms, convergence analysis, and numerical comparisons,” *SIAM journal on scientific and statistical computing*, vol. 9, no. 5, pp. 907–921, 1988.
- [45] P.-A. Absil, R. Mahony, and R. Sepulchre, “Optimization on manifolds: Methods and applications,” in *Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization*. Springer, 2010, pp. 125–144.
- [46] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-manifold preintegration for real-time visual–inertial odometry,” *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.
- [47] J. Sola, J. Dera, and D. Atchuthan, “A micro lie theory for state estimation in robotics,” *arXiv preprint arXiv:1812.01537*, 2018.
- [48] O. Rodrigues, “Des lois géométriques qui régissent les déplacements d’un système solide dans l’espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire,” *Journal de mathématiques pures et appliquées*, vol. 5, pp. 380–440, 1840.

- [49] S. Zambal, W. Palfinger, and C. Eitzinger, “Robotic inspection of 3d cfrp surfaces,” *2016 IEEE Metrology for Aerospace (MetroAeroSpace)*, pp. 197–202, 2016.
- [50] D. Evangelista, M. Terreran, A. Pretto, M. Moro, C. Ferrari, and E. Menegatti, “3d mapping of x-ray images in inspections of aerospace parts,” in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1223–1226.
- [51] A. Pretto, S. Tonello, and E. Menegatti, “Flexible 3d localization of planar objects for industrial bin-picking with monocamera vision system,” in *Proc. of: IEEE International Conference on Automation Science and Engineering (CASE)*, 2013, pp. 168 – 175.
- [52] M. Schwarz, A. Milan, A. S. Periyasamy, and S. Behnke, “Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter,” *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 437–451, 2018.
- [53] R.-J. Halme, M. Lanz, J. Kämäräinen, R. Pieters, J. Latokartano, and A. Hietanen, “Review of vision-based safety systems for human-robot collaboration,” *Procedia CIRP*, pp. 111–116, 2018.
- [54] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco, “Prediction of human activity patterns for human–robot collaborative assembly tasks,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3934–3942, 2019.
- [55] J. Jiang, X. Luo, Q. Luo, L. Qiao, and M. Li, “An overview of hand-eye calibration,” *The International Journal of Advanced Manufacturing Technology*, vol. 119, 03 2022.
- [56] K. Koide and E. Menegatti, “General hand–eye calibration based on reprojection error minimization,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1021–1028, 2019.
- [57] Y. Shiu and S. Ahmad, “Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $ax=xb$,” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 16–29, 1989.
- [58] R. Tsai and R. Lenz, “Real time versatile robotics hand/eye calibration using 3d machine vision,” in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 554–561 vol.1.
- [59] E. Marchand, F. Spindler, and F. Chaumette, “Visp for visual servoing: a generic software platform with a wide class of robot control skills,” *IEEE Robotics Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.

- [60] J. C. K. Chou and M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," *The International Journal of Robotics Research*, vol. 10, no. 3, pp. 240–254, 1991.
- [61] K. Daniilidis and E. Bayro-Corrochano, "The dual quaternion approach to hand-eye calibration," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 1, 1996, pp. 318–322 vol.1.
- [62] K. Strobl and G. Hirzinger, "Optimal hand-eye calibration," in *IEEE International Conference on Intelligent Robots and Systems*, 11 2006, pp. 4647 – 4653.
- [63] F. Park and B. Martin, "Robot sensor calibration: solving $ax=xb$ on the euclidean group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717–721, 1994.
- [64] S. Gwak, J. Kim, and F. Park, "Numerical optimization on the euclidean group with applications to camera calibration," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 65–74, 2003.
- [65] R. Horaud and F. Dornaika, "Hand-eye calibration," *The International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, 1995.
- [66] N. Andreff, R. Horaud, and B. Espiau, "On-line hand-eye calibration," in *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*, 1999, pp. 430–436.
- [67] M. Shah, "Solving the robot-world/hand-eye calibration problem using the kronecker product," *Journal of Mechanisms and Robotics*, vol. 5, no. 3, p. 031007, 2013.
- [68] A. Li, L. Wang, and D. Wu, "Simultaneous robot-world and hand-eye calibration using dual-quaternions and kronecker product," *International Journal of the Physical Sciences*, vol. 5, no. 10, pp. 1530–1536, 2010.
- [69] M. Ulrich and M. Hillemann, "Generic hand–eye calibration of uncertain robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11060–11066.
- [70] Y. Wang, W. Jiang, K. Huang, S. Schwertfeger, and L. Kneip, "Accurate calibration of multi-perspective cameras from a generalization of the hand-eye constraint," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1244–1250.
- [71] D. Evangelista, M. Terreran, A. Pretto, M. Moro, C. Ferrari, and E. Menegatti, "3d mapping of x-ray images in inspections of aerospace parts," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1223–1226.

- [72] F. Dornaika and R. Horaud, “Simultaneous robot-world and hand-eye calibration,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 617–622, 1998.
- [73] J. J. Moré, “The levenberg-marquardt algorithm: implementation and theory,” in *Numerical analysis*. Springer, 1978, pp. 105–116.
- [74] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.
- [75] R. Tsai and R. Lenz, “A new technique for fully autonomous and efficient 3d robotics hand/eye calibration,” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.
- [76] D. Evangelista, D. Allegro, M. Terreran, A. Pretto, and S. Ghidoni, “An unified iterative hand-eye calibration method for eye-on-base and eye-in-hand setups,” in *2022 27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022.
- [77] K. Ebadi, L. Bernreiter, H. Biggie, G. Catt, Y. Chang, A. Chatterjee, C. E. Denniston, S.-P. Deschênes, K. Harlow, S. Khattak *et al.*, “Present and future of slam in extreme environments: The darpa subT challenge,” *IEEE Transactions on Robotics*, vol. 40, pp. 936–959, 2023.
- [78] J. Cheng, L. Zhang, Q. Chen, X. Hu, and J. Cai, “A review of visual slam methods for autonomous driving vehicles,” *Engineering Applications of Artificial Intelligence*, vol. 114, p. 104992, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0952197622001853>
- [79] I. Abaspor Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, “A survey of state-of-the-art on visual slam,” *Expert Systems with Applications*, vol. 205, p. 117734, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417422010156>
- [80] D. Scaramuzza and F. Fraundorfer, “Visual odometry [tutorial],” *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.
- [81] 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 *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1352–1359.
- [82] S. Schubert, P. Neubert, S. Garg, M. Milford, and T. Fischer, “Visual place recognition: A tutorial [tutorial],” *IEEE Robotics and Automation Magazine*, vol. 31, no. 3, pp. 139–153, 2024.

- [83] J. Zhang, S. Singh *et al.*, “Loam: Lidar odometry and mapping in real-time.” in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.
- [84] I. Cvišić, I. Marković, and I. Petrović, “Soft2: Stereo visual odometry for road vehicles based on a point-to-epipolar-line metric,” *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 273–288, 2022.
- [85] F. Dellaert and G. Contributors, “borglab/gtsam,” May 2022. [Online]. Available: <https://github.com/borglab/gtsam>
- [86] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, “g2o: A general framework for (hyper) graph optimization,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 9–13.
- [87] S. Agarwal, K. Mierle, and T. C. S. Team, “Ceres Solver,” 3 2022. [Online]. Available: <https://github.com/ceres-solver/ceres-solver>
- [88] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.
- [89] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [90] X. Gao, R. Wang, N. Demmel, and D. Cremers, “Ldso: Direct sparse odometry with loop closure,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2198–2204.
- [91] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, “Go-slam: Global optimization for consistent 3d instant reconstruction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 3727–3737.
- [92] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, “Appearance-based loop detection from 3d laser data using the normal distributions transform,” in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 23–28.
- [93] M. Bosse and R. Zlot, “Place recognition using keypoint voting in large 3d lidar datasets,” in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 2677–2684.
- [94] C. Cadena, D. Gálvez-López, J. D. Tardós, and J. Neira, “Robust place recognition with stereo sequences,” *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 871–885, 2012.

- [95] D. Rosen and L. Carlone, “Computational enhancements for certifiably correct SLAM,” 2017.
- [96] D. Galvez-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.
- [97] J. Matas and O. Chum, “Randomized ransac with td, d test,” *Image and vision computing*, vol. 22, no. 10, pp. 837–842, 2004.
- [98] J. T. Barron, “A general and adaptive robust loss function,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4331–4339.
- [99] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, “Pairwise consistent measurement set maximization for robust multi-robot map merging,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 2916–2923.
- [100] P. C. Lusk, K. Fathian, and J. P. How, “Clipper: A graph-theoretic framework for robust data association,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 828–13 834.
- [101] J. Neira and J. D. Tardós, “Data association in stochastic mapping using the joint compatibility test,” *IEEE Transactions on robotics and automation*, vol. 17, no. 6, pp. 890–897, 2001.
- [102] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robust map optimization using dynamic covariance scaling,” in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 62–69.
- [103] M. J. Black and A. Rangarajan, “On the unification of line processes, outlier rejection, and robust statistics with applications in early vision,” *International journal of computer vision*, vol. 19, no. 1, pp. 57–91, 1996.
- [104] E. Olson and P. Agarwal, “Inference on networks of mixtures for robust robot mapping,” *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 826–840, 2013.
- [105] M. Ramezani, M. Mattamala, and M. Fallon, “Aeros: Adaptive robust least-squares for graph-based slam,” *Frontiers in Robotics and AI*, vol. 9, p. 789444, 2022.
- [106] D. McGann, J. G. Rogers, and M. Kaess, “Robust incremental smoothing and mapping (risam),” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 4157–4163.

- [107] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “isam2: Incremental smoothing and mapping using the bayes tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.
- [108] N. Sünderhauf and P. Protzel, “Switchable constraints for robust pose graph slam,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1879–1884.
- [109] L. Carlone, A. Censi, and F. Dellaert, “Selecting good measurements via l1 relaxation: A convex approach for robust estimation over graphs,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2667–2674.
- [110] J. Shi, H. Yang, and L. Carlone, “Robin: a graph-theoretic approach to reject outliers in robust estimation using invariants,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 820–13 827.
- [111] Y. Latif, C. Cadena, and J. Neira, “Robust loop closing over time for pose graph slam,” *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.
- [112] F. Wu and G. Beltrame, “Cluster-based penalty scaling for robust pose graph optimization,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6193–6200, 2020.
- [113] E. Olivastri and A. Pretto, “Ipc: Incremental probabilistic consensus-based consistent set maximization for slam backends,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 10 283–10 289.
- [114] G. Grisetti, R. Kümmerle, and K. Ni, “Robust optimization of factor graphs by using condensed measurements,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 581–588.
- [115] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, “Kimera: From slam to spatial perception with 3d dynamic scene graphs,” *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1510–1546, 2021.
- [116] L. Carlone and A. Censi, “From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 475–492, 2014.
- [117] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, “A fast and accurate approximation for planar pose graph optimization,” *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 965–987, 2014.
- [118] N. Sünderhauf and P. Protzel, “Vertigo (versatile extensions for robust inference using graph optimization) source code,” 2019. [Online]. Available: <https://openslam-org.github.io/vertigo.html>

- [119] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, “Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group,” *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 95–125, 2019.
- [120] Z. Zhang and D. Scaramuzza, “A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.
- [121] J. A. Bondy, U. S. R. Murty *et al.*, *Graph theory with applications*. Macmillan London, 1976, vol. 290.
- [122] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [123] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
- [124] K. Khosoussi and I. Shames, “Joint state and noise covariance estimation,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.04584>