TWO NOVEL TECHNIQUES FOR GRAPH OPTIMIZATION

— CYCLE BASED FORMULATION AND CHANGE OF OPTIMAL VALUES

By

FANG BAI

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF TECHNOLOGY SYDNEY
Centre for Autonomous Systems

JAN 2020

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Fang Bai declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy (Engineering) in the Centre for Autonomous Systems at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program, the China Scholarship Council, and the University of Technology Sydney.

Signature:

Production Note:
Signature removed prior to publication.

Date: 27 Jan, 2020

ACKNOWLEDGMENT

I would like to thank Chinese Scholarship Council (CSC) and University of Technology Sydney (UTS) for the adjoint research program that provides me this Ph.D opportunity. I would also like to thank both the school and the research centre for providing the support for multiple oversea travels which greatly broaden my understanding in the research field.

The production of a Ph.D thesis is definitely a teamwork, which requires endeavors and collaborations at all levels. First and foremost, I would like to thank my thesis advisors, Teresa Vidal Calleja and Shoudong Huang, for their patience and spiritual support during the whole research process. I really appreciate that they allow me the maximal freedom all the time that enables me to explore many unorthodox ideas. Besides, I would like to thank my master thesis advisor, Qingling Zhang, for his mentoring and encouragement when taking a PhD program.

I would like to thank many talented researchers I had the privilege to work with. In particular, enormous thanks to Giorgio Grisetti for his mentoring and advice in both life and research at Sapienza University of Rome. I am also greatly honored to work with many brilliant Ph.D students in Rome. Big thanks to Bartolomeo Della Corte, Dominik Schlegel, Irvin Aloise and Lun Wang for the hospitality and broad discussions on various topics. I would like to thank many really smart people in the robotics group I met at Zhejiang University. Big thanks to Rong Xiong, Yue Wang, Li Tang, Xingxing Zuo, Xiaqing Ding, Jixing Lv for their generous help and discussions happened there.

I would like to thank the excellent researchers and postdocs I met in Sydney. Big thanks to Kasra Khosoussi, Jingwei Song, Maani Ghaffari, Cedric Le Gentil, Liang Zhao, Wenjie Lu, Jun Wang, Teng Zhang, Kanzhi Wu, and Yongbo Chen for many inspiring discussions

in terms of both research methodologies and practical research skills. I would like to thank all the friends I met in these four years. In particular, many thanks to Miao Zhang, Jiaheng Zhao, Yanhao Zhang, Brenton Leighton, Tianming Wang, Huan Yu, Raphael Falque, Julian Collart, Daobilige Su, Jiaming Lai, Yin Huan, Runjian Cheng, Yi Liu and etc. for the joy and humor they freely offered during the research process.

At last, I would like to thank my parents, my grandma, my brother, my younger sisters and my uncles, for their ongoing support since I came to this world. None of these would ever happen without you.

TWO NOVEL TECHNIQUES FOR GRAPH OPTIMIZATION

— CYCLE BASED FORMULATION AND CHANGE OF OPTIMAL VALUES

Abstract

by Fang Bai, Ph.D.
University of Technology Sydney
Jan 2020

Graph optimization (GO) is an essential enabling technique widely used in the simultaneous localization and mapping (SLAM), sensor fusion, Lidar based or visual-inertial based navigation systems (VINS). As its name suggests, the graph optimization lies at the intersection of probabilistic inference, sparse linear algebra, and graph theory. In its explicit form, it is a sparse least squares derived from a maximum likelihood estimation (MLE). This thesis contains two fundamental contributions regarding this topic.

A GO can be conventionally represented as a graph with vertices being (unobserved) latent variables, and edges being observed measurements. This vertex-edge paradigm has dominated the GO literature, which in essence solves the problem in the cut space of the graph. In this thesis, we firstly investigate a special GO instance, i.e., pose-graph optimization (PGO), and propose an orthogonal complementary formulation that solves PGO in the cycle space. For sparse graphs, which is typically the case for PGO, the cycle based formulation has a lower dimension of state variables, and takes a form of minimum norm optimization. By exploiting the sparsity by a minimum cycle basis (MCB), the cycle based PGO yields a superior convergence property against its vertex-based counterpart while being cheaper to compute.

The second contribution is the theory on how to forecast the change of optimal values (COOV) in incrementally constructed GO instances. In specific, this thesis develops analytical equations to calculate COOV in case of least squares optimization, and minimum norm optimization. The equation is exactly proved under linear cases, and extends to nonlinear cases via linearizations. We show that COOV bears the same computational complexity as the mutual information (MI) in incremental scenarios, while both COOV and MI well complement one another. As a final contribution, we design several derived applications based on the proposed COOV metric, that demonstrates its effectiveness in outlier detection, cost forecasting, and enhancing the overall robustness in incremental settings. It can be foreseen that numerous applications can be generated based on the two cornerstones in this thesis, and the author would like to leave this part as a future research direction, and open to the community.

TABLE OF CONTENTS

Chapter One

Introduction

The autonomous robot has manifested huge potential in extending human's capabilities, increasing industrial productivities, and preserving lives in the hazardous environment. Indeed, the rapid advancement in robotic applications is transforming the landscape in the field of transportation, manufacturing, medication, disaster rescue, assisting the disabled, infrastructure maintenance, and so forth.

1.1 Motivation

One of the key factor to the robot autonomy is how to model its surrounding environment. This problem is referred to as the perception problem and has been extensively studied in the robotics and computer vision community in the past two decades. A classical instance to this kind is the simultaneous localization and mapping (SLAM) problem, which assumes that a robot is placed in an unknown environment with sensors amounted on the platform, and aims to map its surrounding environment while traveling within it. Both robot executions in the control and observation processes are affected by uncertainties. Obviously, the critical part of SLAM is how to tame the uncertainties. A theoretically principled approach to this task is the probabilistic framework [150], where the uncertainties are modeled in the form of random variables, which are governed by an associated probability distribution reflecting

how likely each state is for the random variable. The method developed in the probabilistic manner benefits from various disciplines, for example probability theory, graph theory, linear algebra tools, and optimization theory etc., thus grounded the foundation of the current state-of-the-art algorithms.

Recent researches show that the perception problem can be largely posed as a probabilistic inference problem (i.e., an estimation problem). An estimation problem is to infer a set of latent variables from a collection of measurements. Taking SLAM as an example, if we let the map being represented by a set of landmarks, and the robot trajectory being a sequence of discretized poses (a state containing both position and rotation component), SLAM can be formulated to solve the position of landmarks and robot poses by using the observation data gathered from the sensors. There exist various estimators for an inference problem, however in robotics (if not in the engineering field) the most popular estimator is the maximum likelihood estimation (MLE). MLE is an unbiased estimator, and turns out to be closely related to the least squares (LS) optimization if the underlying noise is modeled as zero-mean Gaussian. Another key finding to the probabilistic inference is the conditional independence among latent variables, which is often inherent in the stochastic process of the inference problem itself. The conditional independence can be visualized as a graph, i.e., probabilistic graphical models (PGM) [103], with edges in the graph modeling the dependence between latent variables. In many inference instances, for example in SLAM, the PGM is typically sparse. This sparsity is critical to efficient inference solutions because it can be further exploited to take advantage of advanced sparse linear algebra techniques [48]. Since the conditional independence (or equivalently sparsity) is represented by a graph, and the MLE is usually attained by solving a LS optimization, the estimation technique is termed as "graph optimization" (GO) in the community.

GO is crucial because not only it lies in the crux of state estimation, but also it is the foundation of higher level robotic tasks, including but not limited to sensor calibration, localization, pose estimation, map construction, planning, navigation and so forth. There-

fore GO has been a fundamental research topic in robotics, with an abundance of excellent literature extending the boundary of GO in various directions, pivoting around numerical efficiency [52, 92, 95, 96, 105] and robustness [32, 34, 42, 140]. The mainstream researches are grounded on a sparse least squares optimization, with its topology (i.e. sparsity) represented as a vertex-edge paradigm which is a convention in accordance with the PGM. The preceding decade saw a remarkable advancement in this line of research (vertex-edge paradigm). As a result, the state-of-the-art software implementations are capable to solve large GO instances with hundreds and thousands poses in a fraction of seconds, either locally [7, 51, 92, 105] or even globally [140]. In this regards, GO can be largely considered as a solved problem (with various off-the-shelf softwares). However on the other hand, due to its paramount importance, it is always desirable to have more advanced GO techniques, which will undoubtedly benefit the community in various aspects. This thesis contains two fundamental techniques in this regard, which will be motivated in what follows.

The vertex-edge incidence vectors span a space called cut space, whose dimension is decided by the number of vertices. Besides the vertex-edge paradigm, another way to look at the graph topology is based on the cycles in the graph. Typically, a sparse graph implies a much lower dimension in the cycle space (spanned by the independent cycles), compared with that in the cut space. In this regard, this thesis developed a cycle based approach for a particular GO instance, i.e., pose graph optimization (PGO), which is typically rather sparse. The resulting optimization problem takes a form of minimum norm optimization, which is the counterpart of least squares optimization. It is well known in graph theory that the cut space and cycle space are orthogonal complementary. Therefore, in essence, the cycle based approach and vertex based approach are two sides of the same coin. The optimal value of an optimization problem is an important metric to evaluate the quality of the optimal solution. However as a posterior metric, which means the very application of the optimal value relies on the solving of the corresponding problem, the optimal value bears very little practical use, in particular for online (or incremental) applications. To address

this issue, this thesis develops a theory to forecast the change of optimal values between two incrementally constructed GO instances, in the form of both least squares and minimum norm optimization. With this novel theory, we proclaim the optimal value to be a prior (pre-calculated) metric in incremental settings, and it can be foreseen that a large variety of applications can benefit from this new metric.

1.2 Contributions

Concretely, this thesis makes the following contributions:

1. We develop a cycle based PGO framework. We characterize the topology of the graph as a cycle matrix, and re-parameterize the problem using relative poses, which are further constrained by a cycle basis of the graph. We show how the usage of a minimum cycle basis (MCB) improves the computational efficiency, as well as the robustness against local minima. As a byproduct, we design an efficient MCB algorithm incorporating the recent insights in graph theory. We show experimentally that the cycle based PGO has superior convergence properties over its vertex-based counterpart, in terms of both convergence speed and convergence to the global minimum. For sparse graphs that mostly encountered in practice, the cycle-based approach is also more time efficient than the vertex based. We provide open-source C++ implementations for both the cycle based PGO and MCB, to foster further investigations in this regard.

2. We develop the equations to forecast the change of optimal values (COOV) in incremental GO settings, for both least squares optimization (i.e., vertex based GO), and minimum norm optimization (i.e., cycle based GO). We prove the exact equation in the linear case, showing that the change of optimal values is solely decided by the solution and the corresponding covariance in the previously solved problem. We extend the result to nonlinear cases via linearization, featuring optimization instances on manifold.

Therefore the theory to use change of optimal values as a prior (or pre-calculated) metric has been formally established for incrementally constructed GO instances.

3. We develop three typical applications to show how the change of optimal values can dramatically change the methodology in GO studies. The first is to detect outliers in the online setting, by making instant decisions based on the COOV metric. We show experimentally that the proposed method is more robust and much faster than M-estimators. Secondly, we show how the equation to calculate COOV can be used to forecast the optimal cost directly, using an example of aligning two trajectories. The accuracy and computational efficiency are validated with numerical experiments. Last, we show how the COOV metric can be used to design an incremental PGO framework robust against both poor initial values and outliers. The robustness of the proposed framework is demonstrated with experiments on extremely challenging scenarios.

The publications and software implementations that support this thesis are listed as follow:

Journal

† Fang Bai, Teresa Vidal-Calleja, and Shoudong Huang. Robust Incremental SLAM Under Constrained Optimization Formulation. IEEE Robotics and Automation Letters (RA-L), vol. 3, no. 2, April 2018: 1207-1214. (Presented at ICRA 2018, Brisbane.) (link)

† Fang Bai, Teresa Vidal-Calleja, Giorgio Grisetti. Sparse Pose Graph Optimization in Cycle Space. (IEEE Transaction on Robotics (T-RO), conditionally accepted.)

Conference

† Fang Bai, Shoudong Huang, Teresa Vidal-Calleja, and Qingling Zhang. Incremental SQP method for constrained optimization formulation in SLAM. In IEEE International

Conference on Control, Automation, Robotics and Vision (ICARCV), 2016: 1-6. (Best paper award. 1/259.) (link)

† Fang Bai, Teresa Vidal-Calleja, Shoudong Huang, and Rong Xiong. Predicting Objective Function Change in Pose-Graph Optimization. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018. (Presented at Madrid.) (link)

† Fang Bai. Change of Optimal Values: A Pre-calculated Metric. (Accepted at IEEE International Conference on Robotics and Automation (ICRA) 2020, online presentation due to COVID-19.)

C++ Software Implementation

- Cycle Based Pose Graph Optimization (link)

- Outlier Detection in Pose Graph based on the Change of Optimal Values (link)

Chapter Two

Background and Related Work

2.1 Notations

For any two sets $\mathcal{X}_1$ and $\mathcal{X}_2$, we denote respectively $\mathcal{X}_1 \cap \mathcal{X}_2$ the intersection, $\mathcal{X}_1 \cup \mathcal{X}_2$ the union, $\mathcal{X}_1 \backslash \mathcal{X}_2$ the difference, and $\mathcal{X}_1 \oplus \mathcal{X}_2 = (\mathcal{X}_1 \cup \mathcal{X}_2) \backslash (\mathcal{X}_1 \cap \mathcal{X}_2)$ the symmetric difference of these two sets. Let $|\mathcal{X}|$ be the cardinality of the set $\mathcal{X}$. We will use $\mathbb{Z}$ to denote the set of integers, and $\mathbb{R}$ the set of real numbers. If not explicitly stated, the lower-case in normal font, the lower-case in bold font, and the upper-case in bold font are reserved for scalars, vectors, and matrices respectively. A matrix of zeros is denoted by $\mathbf{O}$, and an identity matrix is denoted by $\mathbf{I}$. $\mathbf{A}^T$ represents the transpose, and $\mathbf{A}^\dagger$ the Moore-Penrose pseudo inverse of a matrix $\mathbf{A}$. The notation $\{\mathbf{v}_i\}^\perp$ stands for the orthogonal complement to the space spanned by a set of vectors $\{\mathbf{v}_i\}$. $< \mathbf{v}_1, \mathbf{v}_2 >= \mathbf{v}_1^T \mathbf{v}_2$ is the inner product between $\mathbf{v}_1$ and $\mathbf{v}_2$. The squared Mahalanobis distance is denoted by $\|\mathbf{e}\|_{\mathbf{\Sigma}}^2 = \mathbf{e}^T \mathbf{\Sigma}^{-1} \mathbf{e}$. The notation $[m : n]$ is used to describe a sequence of consecutive integers from m to n. We will use "iff" as a shorthand of "if and only if".

2.2 Preliminaries

2.2.1 Graph Theory

Let $\mathcal{G}$ be an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a finite set, and $\mathcal{E}$ is a set of unordered pairs (u, v), with $u, v \in \mathcal{V}$. The elements in $\mathcal{V}$ are termed vertices (or vertices), and the elements in $\mathcal{E}$ are termed edges. In what follows, we will denote an edge from u to v as e_{uv}. An edge e_{uv} is said to be incident to the vertices u and v, while u and v are called the endpoints of e_{uv}. The degree of a vertex in $\mathcal{G}$ is the number of edges incident to that vertex. A cut-set of $\mathcal{G}$ is a set of edges that separate $\mathcal{V}$ into two disconnected subsets, $\mathcal{V}_1$ and $\mathcal{V}_2$ ($\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$, $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$)), where an edge e_{uv} in the cut-set satisfies $u \in \mathcal{V}_1$, $v \in \mathcal{V}_2$. The set of edges incident to a vertex $v \in \mathcal{V}$ constitute a natural cut-set of $\mathcal{G}$.

A subgraph of $\mathcal{G}$ stands for a graph with only part of vertices and edges from $\mathcal{G}$. In particular, we will be interested in three types of connected subgraphs, i.e., path, cycle, and tree. Formally, a graph is said to be connected if there exists a path for any pair of vertices in the graph. A path is a subgraph in which there are exactly two vertices having degree of one, and the rest of vertices have degree of two. A cycle is a subgraph in which every vertex has an even degree. If the degree of each vertex is exactly two, the cycle is called a simple/elementary cycle, or a circuit. A tree is a subgraph which contains no cycles (i.e. acyclic subgraph). If a tree of $\mathcal{G}$ contains all vertices in $\mathcal{V}$, it is called a spanning tree of $\mathcal{G}$. We will use $\mathcal{P}$ to denote a path, $\mathcal{C}$ a cycle, and $\mathcal{T}$ a tree, respectively.

We will encounter two matrix representations of an undirected graph, (i) incidence matrix $\mathbf{A} = [a_{i,j}]$, with $a_{i,j} = 1$ iff the j-th edge is incident on the i-th vertex, and $a_{i,j} = 0$ otherwise. (ii) cycle matrix $\mathbf{B} = [b_{i,j}]$, with $b_{i,j} = 1$ iff the j-th edge is contained in the i-th cycle, and $b_{i,j} = 0$ otherwise. Since the set of edges incident to a specific vertex is a cut-set of the graph, the incidence matrix is essentially a sub-matrix of the general cut matrix $\mathbf{C} = [c_{i,j}]$, where $c_{i,j} = 1$ iff the j-th edge is contained in the i-th cut-set, and $c_{i,j} = 0$ otherwise. Let the incidence matrix, cycle matrix, and cut matrix of a graph $\mathcal{G}$ be $\mathbf{A}(\mathcal{G})$, $\mathbf{B}(\mathcal{G})$, and $\mathbf{C}(\mathcal{G})$,

respectively. The rows of $\mathbf{A}(\mathcal{G})$, $\mathbf{B}(\mathcal{G})$, and $\mathbf{C}(\mathcal{G})$ span respectively three vector spaces over the two-element finite field based on the arithmetic modulo two. It can be shown that $\mathbf{A}(\mathcal{G})$ and $\mathbf{C}(\mathcal{G})$ have the same row vector space, which is called cut space. Therefore, we consider incidence matrix $\mathbf{A}(\mathcal{G})$ only in later discussion. On the other hand, the rows of $\mathbf{B}(\mathcal{G})$ span a vector space called cycle space. A basis to the cycle space is called cycle basis, which is a complete description of the cycle structure of the graph. For an undirected graph, the cut space and cycle space are orthogonal complement to one other.

A formal description of the cut space and cycle space requires the concept of finite field (or Galois field). A finite field is basically a finite set equipped with arithmetic rules. In particular, we are interested in the finite field of order 2, denoted as $\mathbb{Z}_2 = \{0, 1\}$, whose elements are 0 and 1 only. The addition and multiplication on $\mathbb{Z}_2$ are defined respectively to be the addition and multiplication on $\mathbb{Z}$ modulo 2. Let $|\mathcal{E}| = m$, $|\mathcal{V}| = n$ in $\mathcal{G}$. Then the rows in $\mathbf{A}(\mathcal{G})$, $\mathbf{B}(\mathcal{G})$, and $\mathbf{C}(\mathcal{G})$, can be expressed as m-dimensional vectors on $\mathbb{Z}_2^m$. The row space of $\mathbf{A}(\mathcal{G})$, $\mathbf{B}(\mathcal{G})$, and $\mathbf{C}(\mathcal{G})$ can be accordingly defined based on the vector arithmetics on $\mathbb{Z}_2^m$. For a connected graph $\mathcal{G}$, we can verify that $\mathbf{rank}(\mathbf{A}(\mathcal{G})) = \mathbf{rank}(\mathbf{C}(\mathcal{G})) = n - 1$, $\mathbf{rank}(\mathbf{B}(\mathcal{G})) = m-n+1$, which means that the cut space and cycle space are complementary subspaces of $\mathbb{Z}_2^m$, with the dimension of the cut space being $n - 1$, and the dimension of the cycle space being $m - n + 1$ (also termed cyclomatic number). Another fact derived by the vector arithmetics on $\mathbb{Z}_2^m$ is that $\mathbf{A}(\mathcal{G})\mathbf{B}(\mathcal{G})^T = \mathbf{O}$, $\mathbf{C}(\mathcal{G})\mathbf{B}(\mathcal{G})^T = \mathbf{O}$, which states that the cut space (the row space of $\mathbf{B}(\mathcal{G})$) and cycle space (the row space of $\mathbf{A}(\mathcal{G})$ and $\mathbf{C}(\mathcal{G})$) are orthogonal complementary subspaces over $\mathbb{Z}_2^m$.

A vector $\mathbf{v} \in \mathbb{Z}_2^m$ can be alternatively represented by a subset $\mathcal{S} \subseteq \mathcal{E}$. $\mathcal{S}$ contains the i-th edge iff the i-th element in $\mathbf{v}$ is 1, and do not otherwise. In this case, let $\mathbf{v}_1$, $\mathbf{v}_2$ be two vectors on $\mathbb{Z}_2^m$, and $\mathcal{S}_1$, $\mathcal{S}_2$ be the corresponding set representations. Then the vector addition $\mathbf{v}_1 + \mathbf{v}_2$ on $\mathbb{Z}_2^m$ corresponds to the symmetric difference of sets, i.e., $\mathcal{S}_1 \oplus \mathcal{S}_2$. The inner product satisfies: $< \mathbf{v}_1, \mathbf{v}_2 >= \mathbf{v}_1^T \cdot \mathbf{v}_2 = 1$ iff $|\mathbf{S}_1 \cap \mathbf{S}_2|$ is odd; $< \mathbf{v}_1, \mathbf{v}_2 >= \mathbf{v}_1^T \cdot \mathbf{v}_2 = 0$ iff $|\mathbf{S}_1 \cap \mathbf{S}_2|$ is even. In what follows, we will use the vector representation and set representation

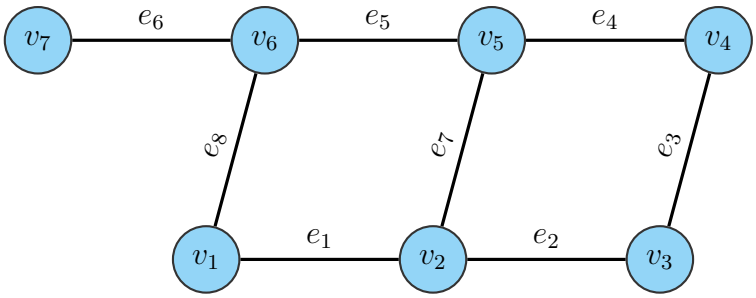


Figure 2.1 A toy undirected graph with 7 vertices and 8 edges.

interchangeably and describe both with the same notation, where the difference can be easily told by the operation used.

We illustrate these concepts with a toy graph in Fig 2.1. The incidence matrix $\mathbf{A}(\mathcal{G})$ and the cycle matrix $\mathbf{B}(\mathcal{G})$ are written explicitly as below.

$$\mathbf{A}(\mathcal{G}) = \begin{array}{c|cccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \\ \hline v_1 & 1 & & & & & & & 1 \\ v_2 & 1 & 1 & & & & & 1 & \\ v_3 & & 1 & 1 & & & & & \\ v_4 & & & 1 & 1 & & & & \\ v_5 & & & & 1 & 1 & & 1 & \\ v_6 & & & & & 1 & 1 & & 1 \\ v_7 & & & & & & 1 & & \end{array}$$

$$\mathbf{B}(\mathcal{G}) = \begin{array}{c|cccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \\ \hline \mathcal{C}_1 & 1 & & & & 1 & & 1 & 1 \\ \mathcal{C}_2 & & 1 & 1 & 1 & & & 1 & \\ \mathcal{C}_3 & 1 & 1 & 1 & 1 & 1 & & & 1 \end{array}$$

To verify the vector arithmetics, let us consider the rows of $\mathbf{B}(\mathcal{G})$, i.e., the cycles $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. The componentwise addition on $\mathbb{Z}_2$ shows that $\mathcal{C}_1 + \mathcal{C}_2 = \mathcal{C}_3$. The set representation of these three cycles are: $\mathcal{C}_1 = \{e_1, e_5, e_7, e_8\}$, $\mathcal{C}_2 = \{e_2, e_3, e_4, e_7\}$, $\mathcal{C}_3 = \{e_1, e_2, e_3, e_4, e_5, e_8\}$. The symmetric difference between two sets yields $\mathcal{C}_1 \oplus \mathcal{C}_2 = \mathcal{C}_3$, which is in accordance with $\mathcal{C}_1 + \mathcal{C}_2 = \mathcal{C}_3$. Therefore $\mathcal{C}_3$ can be expressed by $\mathfrak{B} = \{\mathcal{C}_1, \mathcal{C}_2\}$, which is a cycle basis of the

graph. The inner product between $\mathcal{C}_1$ and $\mathcal{C}_2$ writes $< \mathcal{C}_1, \mathcal{C}_2 >= \mathcal{C}_1^T \cdot \mathcal{C}_2 = 1$, while we verify that $|\mathcal{C}_1 \cap \mathcal{C}_2|$ is odd. The orthogonality between a row in $\mathbf{A}(\mathcal{G})$ and a row in $\mathbf{B}(\mathcal{G})$ can be verified accordingly.

2.2.2 Lie Group

We will encounter two Lie groups in later discussion, the special orthogonal group SO(3) and special Euclidean group SE(3). Both SO(3) and SE(3) are matrix Lie groups, which are closed under matrix multiplications. The identity of the group is the identity matrix.

SO(3) **and** SE(3)

SO(3), also termed rotational group, is a collection of valid rotation matrices (in the right-hand coordinate frame), that formally writes

$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} \ : \ \mathbf{R}^T\mathbf{R} = \mathbf{I}, \ \mathbf{det}(\mathbf{R}) = 1\}.$$

SE(3) is a semi-direct product of SO(3) and $\mathbb{R}^3$, i.e., $(\mathrm{SE}(3), \circ) = (\mathbb{R}^3, +) \rtimes (\mathrm{SO}(3), \cdot)$, with its group operation $\circ$ defined as

$$(\mathbf{t}_1, \ \mathbf{R}_1) \circ (\mathbf{t}_2, \ \mathbf{R}_2) = (\mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1, \ \mathbf{R}_1\mathbf{R}_2), \quad \forall \ \mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^3, \ \mathbf{R}_1, \mathbf{R}_2 \in \mathrm{SO}(3).$$

The group operation $\circ$ for SE(3) can be compactly expressed as matrix multiplications, by writing $(\mathbf{t}, \mathbf{R})$ as a matrix $\mathbf{T} \in \mathbb{R}^{4\times 4}$ in the form

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0} & 1 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0} & 1 \end{bmatrix} \quad \longrightarrow \quad \mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1 \\ \mathbf{0} & 1 \end{bmatrix}.$$

Obviously, $(\mathbf{t}_1, \mathbf{R}_1) \circ (\mathbf{t}_2, \mathbf{R}_2)$ corresponds to the multiplication $\mathbf{T}_1\mathbf{T}_2$. Therefore we formally define SE(3) as a matrix group

$$\mathrm{SE}(3) = \{\mathbf{T} \in \mathbb{R}^{4\times 4} \ : \ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \ \mathbf{R} \in \mathrm{SO}(3), \ \mathbf{t} \in \mathbb{R}^3\}.$$

Tangent Space

Both SE(3) and SO(3) form smooth manifolds, embedded in the vector space, i.e., $\mathbb{R}^{3\times3}$ and $\mathbb{R}^{4\times4}$, respectively. The tangent space to such manifolds (i.e., embedded sub-manifolds in the vector space) can be identified by using smooth curves on the manifold [4]. Let $\mathcal{M}$ be a sub-manifold embedded in a vector space. A curve in $\mathcal{M}$ is a smooth mapping: $\gamma : \mathbb{R} \rightarrowtail \mathcal{M} : t \rightarrowtail \gamma(t)$. Let $\gamma(0) = x \in \mathcal{M}$. Then the tangent space of $\mathcal{M}$ at the point $x \in \mathcal{M}$, denoted by $T_x\mathcal{M}$, is identified by the set:

$$T_x\mathcal{M} = \{\frac{\partial\gamma(t)}{\partial t}\Big|_{t=0} : \gamma(t) \text{ is a curve in } \mathcal{M},\ \gamma(0) = x\} \tag{2.1}$$

where $\frac{\partial\gamma(t)}{\partial t}\Big|_{t=0}$ is called a tangent vector at x realized by the curve $\gamma(t)$.

Let us examine the tangent space of SO(3) based on (2.1). Let $\mathbf{X}(t)$ be a curve in SO(3), with $\mathbf{X}(0) = \mathbf{R}$. Since $\mathbf{X}(t)$ lies in SO(3), we have

$$\mathbf{X}(t)^T\mathbf{X}(t) = \mathbf{I}.$$

Differentiating both sides with respect to t yields,

$$\left(\frac{\partial\mathbf{X}(t)}{\partial t}\right)^T\mathbf{X}(t) + \mathbf{X}(t)^T\left(\frac{\partial\mathbf{X}(t)}{\partial t}\right) = \mathbf{O}.$$

At the point $\mathbf{X}(0) = \mathbf{R}$, we deduce that $\mathbf{Z} = \frac{\partial\mathbf{X}(t)}{\partial t}\Big|_{t=0}$ belongs to the set

$$T_{\mathbf{R}\in\mathrm{SO}(3)} = \{\mathbf{Z} \in \mathbb{R}^{3\times3} \mid \mathbf{R}^T\mathbf{Z} + \mathbf{Z}^T\mathbf{R} = \mathbf{O}\}$$

which literally characterizes the tangent space of SO(3) at an arbitrary $\mathbf{R} \in \mathrm{SO}(3)$.

Lie Algebra

For Lie groups, we are particularly interested in the tangent space at the identity of the group (i.e., an identity matrix $\mathbf{I}$):

$$\mathfrak{so}(3) = T_{\mathbf{I}\in\mathrm{SO}(3)} = \{\boldsymbol{\Omega} \in \mathbb{R}^{3\times3} \mid \boldsymbol{\Omega} + \boldsymbol{\Omega}^T = \mathbf{O}\}. \tag{2.2}$$

The set $\mathfrak{so}(3)$ in (2.2) is essentially a set of skew-symmetric matrices, which satisfy $\boldsymbol{\Omega}^T = -\boldsymbol{\Omega}^T$. A skew-symmetric matrix $\boldsymbol{\Omega} \in \mathbb{R}^{3\times 3}$ has a peculiar structure:

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}^\wedge = \boldsymbol{\omega}^\wedge$$

which can be characterized by a 3-dimensional vector $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T \in \mathbb{R}^3$. We will use $\wedge$ and $\vee$ to describe the relationship between $\boldsymbol{\Omega}$ and $\boldsymbol{\omega}$: $\boldsymbol{\Omega} = \boldsymbol{\omega}^\wedge$, $\boldsymbol{\omega} = \boldsymbol{\Omega}^\vee$.

Analogously, we characterize the tangent space of SE(3) at the identity of the group to be $\mathfrak{se}(3) = T_{\mathbf{I}\in \mathrm{SE}(3)} = \{\boldsymbol{\Xi}\}$. The matrix $\boldsymbol{\Xi}$ is termed a "screw matrix" taking the form:

$$\boldsymbol{\Xi} = \begin{bmatrix} 0 & -\xi_3 & \xi_2 & \xi_4 \\ \xi_3 & 0 & -\xi_1 & \xi_5 \\ -\xi_2 & \xi_1 & 0 & \xi_6 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \\ \xi_5 \\ \xi_6 \end{bmatrix}^\wedge = \boldsymbol{\xi}^\wedge.$$

Obviously $\boldsymbol{\Xi}$ can be characterized by a 6-dimensional vector $\boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6]^T \in \mathbb{R}^6$, and we analogously express the relation between $\boldsymbol{\Xi}$ and $\boldsymbol{\xi}$ to be: $\boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge$, $\boldsymbol{\xi} = \boldsymbol{\Xi}^\vee$.

The Lie bracket $[\cdot, \cdot]$ is a binary operator defined as

$$[\mathbf{X}, \mathbf{Y}] \coloneqq \mathbf{XY} - \mathbf{YX}.$$

Both the set $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$ are closed under the Lie bracket, i.e., if $\mathbf{X}, \mathbf{Y} \in \mathfrak{so}(3)$, then $[\mathbf{X}, \mathbf{Y}] \in \mathfrak{so}(3)$; likewise if $\mathbf{X}, \mathbf{Y} \in \mathfrak{se}(3)$, then $[\mathbf{X}, \mathbf{Y}] \in \mathfrak{se}(3)$. Therefore, both $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$ are in essence vector spaces endowed with an algebraic structure, which are termed the Lie algebra of SO(3) and SE(3) respectively. The Lie bracket is sometimes called the commuter, which can be interpreted as a measure on the matrix commutation. If $\mathbf{X}$ and $\mathbf{Y}$ commute, $[\mathbf{X}, \mathbf{Y}] = \mathbf{O}$. The Lie bracket is useful when describing the Baker–Campbell–Hausdorff (BCH) formula which we will introduce later on.

In what follows, we will recall some commonly used properties on Lie group. It is worth noting that most of these properties stand for general Lie groups, i.e., for both SO(3) and SE(3), and other matrix Lie groups like sim(3) and etc.. We will use G to represent a general Lie group, and $\mathfrak{g}$ its Lie algebra.

Exponential and Logarithm Mapping

A matrix Lie group and its Lie algebra are connected by matrix exponentials. Let $\mathbf{x}^\wedge \in \mathfrak{g}$, then there exists an element $\mathbf{X} \in \mathrm{G}$ such that

$$\mathbf{exp} \ : \ \mathfrak{g} \rightarrowtail \mathrm{G} \ : \ \mathbf{X} = \mathbf{exp}(\mathbf{x}^\wedge).$$

The matrix exponential **exp** brings an element in $\mathfrak{g}$ to an element in G, thus in Lie group theory it is called exponential mapping from $\mathfrak{g}$ to G. The exponential mapping is surjective, which means for any $\mathbf{X} \in \mathrm{G}$, we can find (at least) one $\mathbf{x}^\wedge \in \mathfrak{g}$ that realizes $\mathbf{X} = \mathbf{exp}(\mathbf{x}^\wedge)$. It is possible that different $\mathbf{x}_1^\wedge, \mathbf{x}_2^\wedge \in \mathfrak{g}$ maps to exactly the same element $\mathbf{X} \in \mathrm{G}$. However, we can always confine ourselves to a subset of $\mathfrak{g}$ (for example, with minimal vector norm), and define an injective mapping by matrix logarithmic,

$$\mathbf{log} \ : \ \mathrm{G} \rightarrowtail \mathfrak{g} \ : \ \mathbf{x} = \mathbf{log}(\mathbf{X})^\vee$$

which is termed logarithm mapping from G to $\mathfrak{g}$.

Although both the matrix exponential **exp**() and logarithmic **log**() are defined by Taylor series, for most Lie groups of interests, like SO(3) and SE(3), we can work out a closed form expression for $\mathbf{exp}(\cdot)$ and $\mathbf{log}(\cdot)$ [19, 45]. For the clarity of notations, we define the encapsulated exponential and logarithm mapping, $\mathbf{Exp} \ : \ \mathbb{R}^n \rightarrowtail \mathrm{G}$, and $\mathbf{Log} \ : \ \mathrm{G} \rightarrowtail \mathbb{R}^n$:

$$\mathbf{Exp} \ : \ \mathbb{R}^n \rightarrowtail \mathrm{G} \ : \ \mathbf{X} = \mathbf{Exp}(\mathbf{x}) = \mathbf{exp}(\mathbf{x}^\wedge)$$

$$\mathbf{Log} \ : \ \mathrm{G} \rightarrowtail \mathbb{R}^n \ : \ \mathbf{x} = \mathbf{Log}(\mathbf{X}) = \mathbf{log}(\mathbf{X})^\vee.$$

Adjoint

It can be easily verified by examining the Taylor series of a matrix exponential that $\mathbf{exp}(\cdot)$ satisfies the following identity:

$$\mathbf{A}\mathbf{exp}(t\boldsymbol{\Omega})\mathbf{A}^{-1} = \mathbf{exp}(t\mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1}). \tag{2.3}$$

Let us consider the case that $\boldsymbol{\Omega} \in \mathfrak{g}$, $\mathbf{A} \in \mathrm{G}$. Obviously $t\boldsymbol{\Omega} \in \mathfrak{g}$ and $\mathbf{exp}(t\boldsymbol{\Omega}) \in \mathrm{G}$. Since G is closed, we have $\mathbf{A}\mathbf{exp}(t\boldsymbol{\Omega})\mathbf{A}^{-1} \in \mathrm{G}$, implying $\mathbf{exp}(t\mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1}) \in \mathrm{G}$. As a result, $\gamma(t) = \mathbf{exp}(t\mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1})$ is a curve in G, with $\gamma(0) = \mathbf{I}$. Differentiating $\gamma(t)$ at $t = 0$, we get,

$$\frac{\partial \gamma(t)}{\partial t}\Big|_{t=0} = \frac{\partial \mathbf{exp}(t\mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1})}{\partial t}\Big|_{t=0} = \mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1}$$

which is a tangent vector at the identity of the Lie group. Therefore $\mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1} \in \mathfrak{g}$.

The adjoint operation at $\mathbf{A} \in \mathrm{G}$ is a mapping $\mathbf{Adj}_{\mathbf{A}} : \mathfrak{g} \mapsto \mathfrak{g}$ that writes:

$$\mathbf{Adj}_{\mathbf{A}} : \mathfrak{g} \mapsto \mathfrak{g} : \mathbf{Adj}_{\mathbf{A}}(\boldsymbol{\Omega}) \coloneqq \mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1}.$$

It can be shown that $\mathbf{Adj}_{\mathbf{A}}(\cdot)$ is linear, since $\mathbf{Adj}_{\mathbf{A}}(a\boldsymbol{\Omega}_1 + b\boldsymbol{\Omega}_2) = a\mathbf{Adj}_{\mathbf{A}}(\boldsymbol{\Omega}_1) + b\mathbf{Adj}_{\mathbf{A}}(\boldsymbol{\Omega}_2)$, $\forall\ \boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2 \in \mathfrak{g}$. Therefore there exists a matrix $\mathbf{Ad}_{\mathbf{A}}$ such that $\mathbf{Adj}_{\mathbf{A}}(\boldsymbol{\Omega}) = (\mathbf{Ad}_{\mathbf{A}} \cdot \boldsymbol{\omega})^{\wedge}$, with $\boldsymbol{\Omega} = \boldsymbol{\omega}^{\wedge}$. To accommodate different matrix $\mathbf{A}$, we write $\mathbf{Ad}_{\mathbf{A}} = \mathbf{Ad}(\mathbf{A})$:

$$\mathbf{Adj}_{\mathbf{A}}(\boldsymbol{\Omega}) = \mathbf{A}\boldsymbol{\Omega}\mathbf{A}^{-1} = (\mathbf{Ad}_{\mathbf{A}} \cdot \boldsymbol{\omega})^{\wedge} = (\mathbf{Ad}(\mathbf{A}) \cdot \boldsymbol{\omega})^{\wedge}.$$

It can be shown that $\mathbf{Ad}(\mathbf{AB}) = \mathbf{Ad}(\mathbf{A})\mathbf{Ad}(\mathbf{B})$, $\forall\ \mathbf{A}, \mathbf{B} \in \mathrm{G}$. For SO(3) and SE(3), the adjoint operation $\mathbf{Ad}(\cdot)$ is given in closed form [19, 45].

By (2.3), we derive a useful property regarding the matrix exponential and adjoint:

$$\begin{gathered}\mathbf{A}\mathbf{exp}(\boldsymbol{\omega}^{\wedge})\mathbf{A}^{-1} = \mathbf{exp}((\mathbf{Ad}(\mathbf{A}) \cdot \boldsymbol{\omega})^{\wedge}) \\ \iff \mathbf{A}\mathbf{Exp}(\boldsymbol{\omega})\mathbf{A}^{-1} = \mathbf{Exp}(\mathbf{Ad}(\mathbf{A}) \cdot \boldsymbol{\omega}),\ \forall\ \mathbf{A} \in \mathrm{G},\ \boldsymbol{\omega}^{\wedge} \in \mathfrak{g}\end{gathered}$$

which can be used to shift the position between $\mathbf{A}$ and $\mathbf{Exp}(\cdot)$.

Baker–Campbell–Hausdorff (BCH) Formula

Matrix multiplications are not commutative, namely for two general matrices $\mathbf{A}$, $\mathbf{B}$, we get $\mathbf{AB} \neq \mathbf{BA}$. Equivalently it is said $[\mathbf{A}, \mathbf{B}] \neq \mathbf{O}$, where $[\cdot, \cdot]$ is the Lie bracket introduced before. If $\mathbf{A}$ and $\mathbf{B}$ are not commutative, their matrix exponentials cannot be concatenated directly, which formally writes:

$$\mathbf{exp(A)exp(B)} \neq \mathbf{exp(A+B)}, \text{ if } [\mathbf{A}, \mathbf{B}] \neq \mathbf{O}.$$

In this case, the multiplication $\mathbf{exp(A)exp(B)}$ still admits a form of matrix exponential, i.e., $\mathbf{exp(A)exp(B)} = \mathbf{exp(C)}$, with $\mathbf{C}$ expressed as a series with respect to $\mathbf{A}$ and $\mathbf{B}$. This series is called Baker–Campbell–Hausdorff (BCH) formula, whose first four orders are shown as below:

$$\mathbf{C} = \mathbf{A} + \mathbf{B} + \frac{1}{2}[\mathbf{A}, \mathbf{B}] + \frac{1}{12}([\mathbf{A}, [\mathbf{A}, \mathbf{B}]] + [\mathbf{B}, [\mathbf{B}, \mathbf{A}]]) - \frac{1}{24}[\mathbf{B}, [\mathbf{A}, [\mathbf{A}, \mathbf{B}]]] + \cdots .$$

We are particularly interested in the case that $\mathbf{A}, \mathbf{B}, \mathbf{C}$ belongs to a Lie algebra. In general we cannot work out a closed form for the BCH formula, even for commonly used Lie groups like SO(3) and SE(3). A compromise is to seek for approximations. Below is a commonly used approximation that holds for both SO(3) and SE(3). Let $\mathbf{x}^\wedge \in \mathfrak{g}$, $\mathbf{y}^\wedge \in \mathfrak{g}$. We expressed the result with encapsulated $\mathbf{Exp}(\cdot)$ and $\mathbf{Log}(\cdot)$:

$$\mathbf{Exp}(\mathbf{x}) \cdot \mathbf{Exp}(\mathbf{y}) \approx \begin{cases} \mathbf{Exp}\left(\mathbf{J}_l^{-1}(\mathbf{y})\,\mathbf{x} + \mathbf{y}\right), & \text{if } \mathbf{x} \to \mathbf{0} \\ \mathbf{Exp}\left(\mathbf{x} + \mathbf{J}_r^{-1}(\mathbf{x})\,\mathbf{y}\right), & \text{if } \mathbf{y} \to \mathbf{0} \end{cases}$$

where $\mathbf{J}_l(\cdot)$, $\mathbf{J}_r(\cdot)$ are called respectively the left-hand Jacobian and right-hand Jacobian of the exponential coordinate parameterization. Both these Jacobians and their inversions are calculated in closed form [19, 45] for SO(3) and SE(3).

2.3 Graph Optimization

In this section, we will revisit the basics on graph optimization in the context of this thesis, which is an interplay between graph theory and least squares optimization. The graph is used to describe the independence in the probabilistic formulation of the problem, and encode the sparsity pattern in the numerical calculation thereafter. In robotics, the technique is termed as SLAM back-end optimization [31], which draws extensive connections with the probabilistic inference [103], and estimation theory [19, 114]. This chapter provides the key concepts and literature regarding this topic. Interested readers are encouraged to consult the excellent material in [17, 19, 31, 54, 61, 103, 114] for a broader understanding.

2.3.1 Maximum Likelihood Estimation

Many robotic applications require to estimate a set of (unobserved) variables using a collection of observed measurements. Taking SLAM as an example. A robot travels inside an unknown environment. The robot collects information of its surroundings by on board sensors. Then an estimation problem arises which aims to obtain a (maximal consistent) configuration of robot poses and the map by using the collected measurements. A systematic approach to estimation problems of this kind is based on a probabilistic framework [150].

Let $\mathbf{x}$ be the (unobserved) variables which needs to be estimated, i.e. the set of robot poses and the map representation in SLAM. Let us denote $\mathbf{z}$ as the observed variable, and $\tilde{\mathbf{z}}$ be the corresponding measurements to $\mathbf{z}$. In SLAM, $\mathbf{z}$ stands for a set of relative measurements of the environment inside the robot's local coordinate frame. The relative measurements can take various forms, depending on the sensor used, for example the image captured by a camera, the range/bearing of obstacles reported by a Lidar, the angular velocity and translational acceleration obtained by an Inertial Measurement Unit (IMU), etc.

In the probabilistic sense, the measurement $\tilde{\mathbf{z}}$ can be interpreted as a random realization (i.e., a sample) of the conditional probability $P(\mathbf{z}|\mathbf{x})$, which is also termed the likelihood of

$\mathbf{z}$ given the unobserved latent variable $\mathbf{x}$. The maximum likelihood estimates (MLE) is to obtain an estimate $\hat{\mathbf{x}}$ for $\mathbf{x}$ which maximizes the likelihood,

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}} \ P(\mathbf{z} = \tilde{\mathbf{z}}|\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}} \ -\log P(\mathbf{z} = \tilde{\mathbf{z}}|\mathbf{x}). \tag{2.4}$$

The second equality holds because $\log(\cdot)$ is monotone. It turns out computing $\log P(\mathbf{z}|\mathbf{x})$ can be much easier than $P(\mathbf{z}|\mathbf{x})$ itself, especially when $P(\cdot)$ belongs to an exponential family.

2.3.2 Conditional Independence and Graphical Models

It can be foreseen that $P(\mathbf{z}|\mathbf{x})$ takes a high dimensional complex distribution, thus a direct deployment of (2.4) is both analytically and computationally intractable. To mitigate this issue, we can further exploit the inherent conditional independence, by making minor assumptions (i.e., Markov assumption) based on the specific structure of the problem. Basically, we aim to factorize the high dimensional complex probability $P(\mathbf{z}|\mathbf{x})$ into a product of low dimensional simple probabilities $P_i(\mathbf{z}_i|\mathbf{x}_i)$ as below,

$$P(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{m} P_i(\mathbf{z}_i|\mathbf{x}_i) \tag{2.5}$$

where $P_i(\mathbf{z}_i|\mathbf{x}_i)$ $(i = 1, \ldots, m)$ are independent from one another, with $\mathbf{z}_i$ and $\mathbf{x}_i$ containing only several components from $\mathbf{z}$ and $\mathbf{x}$ respectively.

The conditional independence arises because the underlying stochastic process is assumed to be Markov [150]. In plain words, the Markov assumption states that the latent variable is memoryless, so that the current state is sufficient to make future decisions. The probabilistic independence can be easily described by the so-called probabilistic graphical models [103]. Fig. 2.2 is a Bayesian network or commonly known as Bayes net describing the stochastic process of SLAM. In the Bayes net, a directed edge $x \to y$ is created if the variable y is conditionally dependent on x. In Fig. 2.2, the current state of the robot $\mathbf{x}_i$, and the forthcoming control input $\mathbf{u}_i$ are sufficient to determine the future state of the robot $\mathbf{x}_{i+1}$, writing a conditional probability $P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i)$. In other words, given the current state of

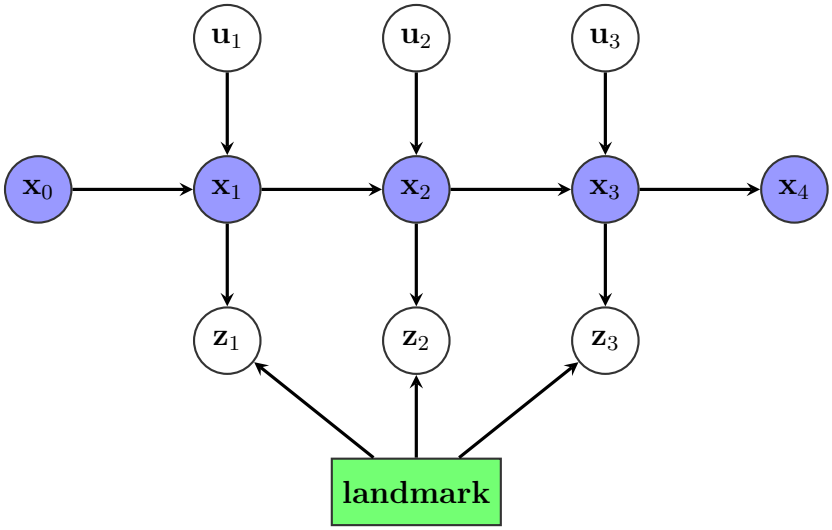


Figure 2.2 A Bayes net representation of a typical SLAM problem.

the robot $\mathbf{x}_i$, the future state $\mathbf{x}_{i+1}$ is independent of the past robot state $\mathbf{x}_{i-1}$. Another example is that the observation to a static landmark is only dependent on the robot pose where the observation is made, for example, an observation $\mathbf{z}_i$ is only dependent on $\mathbf{x}_i$ and the landmark position. We write this conditional probability as $P(\mathbf{z}_i|\mathbf{x}_i)$. Intuitively speaking, the probabilistic graphical model represent variables by vertices in the graph, and describe the conditional dependence among vertices by the edges. This "vertex-edge" paradigm is the foundation of the mainstream SLAM research since it was brought into the robotics community.

While Bayes Net is a powerful tool to model and visualize the stochastic process, it is less convenient in the optimization perspective. Taking the state transition model $P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i)$ in Fig 2.2 as an example. We notice that the variable $\mathbf{x}_{i+1}$ on the left-hand side of the conditional symbol "|" is unobserved, which contradicts the form of factorization in (2.5). To mitigate this issue, we can reformulate $P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i)$ as $P(\mathbf{u}_i|\mathbf{x}_i, \mathbf{x}_{i+1})$, and edit the structure of Bayes Net accordingly. Alternatively, we can keep Bayes Net unchanged for the stochastic process (which is quite useful for filtering methods [57]), and introduce another graphical model for this purpose to benefit the optimization perspective. A commonly used graphical model to this regard is the so-called factor graph [54].

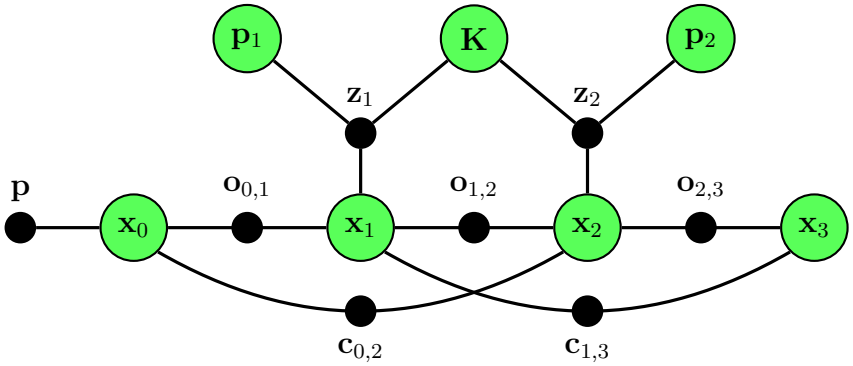


Figure 2.3 A factor graph representation of a visual SLAM system.

A factor graph is a bipartite graph, with two different types of vertices: factor nodes, and variable nodes. See Fig. 2.3 for a toy example of the factor graph representation, in which the large filled circles represent variable nodes, and black dots stand for factor nodes. Typically, the variable nodes are latent variables to be inferred, and factor nodes are observed variables (i.e., measurements). Edges are always between the factor nodes and variable nodes, modeling the conditional independence between them. In factor graphs, factors nodes are always conditioned on the variable nodes, therefore the "arrow" (i.e., orientation) is neglected for edges. The overall factor graph represents a factorization of the conditional probability density function described in (2.5). For a thorough exposition on this topic, we refer interested readers to [54].

2.3.3 Least Squares Optimization

Last but not least, to reduce the probabilistic framework to calculations, we need to know an explicit expression for $P_i(\mathbf{z}_i|\mathbf{x}_i)$. To that end, we have to assign a measurement model, and a noise model. A measurement model is typically obtained by analyzing the physical mechanisms of the underlying problem, which writes a function mapping the unobserved latent variable $\mathbf{x}$ to the observed variable $\mathbf{z}$,

$$\mathbf{z}_i = \mathbf{m}_i(\mathbf{x}_i).$$

While the measurement model itself is deterministic, i.e, noise free, the real measurement data obtained from sensor readings are inevitably corrupted by noise. A noise model is a probability distribution aiming to model the uncertainty occurred during the measurement process. The most popular unimodal noise model is attributed to the zero-mean Gaussian, which is simple (characterized by mean and variance) and easy to compute (being an exponential family), and proved to work well in practice. In general, we assume the noise resides in the tangent space of the measurement, which writes,

$$\mathbf{m}_i(\mathbf{x}_i) \boxminus \mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_i). \tag{2.6}$$

The notation $\boxminus$ in (2.6) is a generalization of the Euclidean difference to manifold.

By (2.6), based on the probability density function of Gaussian, we can write down an explicit expression for $P_i(\mathbf{z}_i|\mathbf{x}_i)$ which takes the form

$$P_i(\mathbf{z}_i|\mathbf{x}_i) \propto c \exp\left(-\frac{1}{2}\left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \mathbf{z}_i\right)^T \mathbf{\Sigma}_i^{-1} \left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \mathbf{z}_i\right)\right). \tag{2.7}$$

Here c is a normalizer, i.e., a constant, that ensures the probability density integrates to 1. It is worth noting that the probability density function (2.7) is not necessarily a Gaussian. In particular, the resulting distribution is banana shaped [20, 111] when dealing with SO(3) or SE(3) groups. Since the distribution is driven by the Gaussian in the tangent space, see (2.6), it is called "concentrated Gaussian" in some literatures [28, 154].

By (2.7) (2.5) (2.4), it can be further shown that the MLE boils down to a least squares optimization problem:

$$\begin{aligned}
\hat{\mathbf{x}} &= \underset{\mathbf{x}}{\operatorname{argmin}} \; -\log \prod_{i=1}^{m} P_i(\mathbf{z}_i = \tilde{\mathbf{z}}_i|\mathbf{x}_i) \\
&= \underset{\mathbf{x}}{\operatorname{argmin}} \; -\sum_{i=1}^{m} \log P_i(\mathbf{z}_i = \tilde{\mathbf{z}}_i|\mathbf{x}_i) \\
&= \underset{\mathbf{x}}{\operatorname{argmin}} \; \sum_{i=1}^{m} \left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \tilde{\mathbf{z}}_i\right)^T \mathbf{\Sigma}_i^{-1} \left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \tilde{\mathbf{z}}_i\right) + \text{const} \\
&= \underset{\mathbf{x}}{\operatorname{argmin}} \; \sum_{i=1}^{m} \left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \tilde{\mathbf{z}}_i\right)^T \mathbf{\Sigma}_i^{-1} \left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \tilde{\mathbf{z}}_i\right)
\end{aligned} \tag{2.8}$$

Proposition 1 (MLE by Least Squares)**.** A maximum likelihood estimate $\hat{\mathbf{x}}$, for the variable $\mathbf{x}$ can be obtained by solving a least squares optimization problem:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{m} \left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \tilde{\mathbf{z}}_i\right)^T \boldsymbol{\Sigma}_i^{-1} \left(\mathbf{m}_i(\mathbf{x}_i) \boxminus \tilde{\mathbf{z}}_i\right). \tag{2.9}$$

The graph optimization is to obtain a MLE by solving the least squares optimization in (2.9), by efficiently exploiting the sparsity represented by a graph (i.e., factor graph).

2.4 Related Work

GO lies at the intersection of several disciplines, i.e., estimation theory, numerical optimization, graph theory, and sparse linear algebra, and is a fundamental tool in applications like SLAM. Therefore there exists an extensive amount of literature regarding this topic. In what follows, we mainly focus on the development of GO in robotics. Readers interested in a broader exposition on the topic are encouraged to consult [25, 107, 115] for least squares optimization, [54, 103, 150] for graphical models, [31, 147] for SLAM, [48] for sparse linear algebra, and the tutorial paper [77, 124] for MLE and GO.

Maximum Likelihood Estimation and Graph Optimization

In robotics, Lu et al. [113] was the first to formulate SLAM as a MLE, where a nonlinear least squares (NLS) is used to optimize the network generated by scan-matchings. At the time, although in theory, techniques like Gauss-Newton [115] were available to solve the NLS, the development of its numerical side was a bit behind. To address the computational complexity, Frese et al. [68] proposed a multi-level relaxation method, based on the Gauss-Seidel relaxation. Olson et al. [127] suggested a stochastic gradient descent method to solve the NLS. Combined with a relative parameterization, this method yielded a large basin of convergence to the global optimum, while being substantially faster than the Gauss-Seidel based method. Grisetti et al. [76] extended the framework to 3D, and applied a tree parameterization to improve the convergence speed.

The rapid advancement in numerical computations, in particular the sparse graphical techniques to solve large scale sparse linear systems [48], completely changed the landscape. In robotics, Dellaert et al. [52] is the first to show that MLE, i.e., NLS, can be solved efficiently by a Gauss-Newton method, using sparse matrix decompositions. Kaess et al. attributed to the incremental solver of MLE, using the Givens rotation based QR decomposition [95], or the Bayes tree [96]. Kummerle et al. [105] designed a general graph based optimization framework. Ila et al. [92] exploited the block structure of sparse matrices. In general, the Gauss-Newton method has a much faster convergence rate than gradient based methods, and the approximation in Hessian results in a specific sparsity pattern that can be exploited with sparse linear algebraic techniques, which forms the basis of the state-of-the-art GO solvers [7, 51, 92, 105].

Besides direct sparse matrix decompositions [48], the linear system resulted from the Gauss-Newton based methods can also be solved by an iterative method, for example preconditioned conjugate gradient [53, 104, 105, 121]. As a standard optimization technique, the global convergence of the Gauss-Newton method [65] can be further improved by using trust region techniques [115], like Levenberg-Marquardt [116] and Powell's-dog-leg [137]. All these techniques can be seamlessly integrated into existing GO libraries.

SLAM and Pose Graph Optimization

For some GO instances we are particularly interested in, like PGO which typically arises in SLAM, it is possible to design more sophisticated solvers with minor adaption on the general GO framework. In what follows, we review the major insights into PGO, and refer readers interested in a thorough exposition to SLAM to the excellent texts [17, 31, 61, 147, 150].

Grisetti et al. [78, 79] proposed the divide-and-conquer methods. The basic idea is to divide the full PGO into several submaps (i.e. subgraphs), solve each of them, and then join the submaps together to obtain an approximation to the full PGO. Zhao et al. [159, 160] investigated the special case of joining two submaps, with a clever parameterization which

can be solved by a linear least squares, followed by a nonlinear transformation. For 2D cases, Carlone et al. [38] suggested a linear approximation framework to PGO, by computing firstly an orientation estimation, and then the position part using the given orientation. The core was the regularization of rotation angles [38], which was systematically addressed in [34] using a quadratic integer programming. The separability of the orientation and position estimation was further studied by Khosoussi et al. [101], based on a variable projection approach.

Besides practical algorithms to solve PGO, some theoretical insights are also drawn. Huang et al. [87] showed empirically that a point-feature based SLAM is close to a convex optimization problem, and a relative formulation is proposed for the purpose of reducing the nonlinearity in SLAM. Wang et al. [152] discussed the number of local minimums for PGO in special cases. Carlone [32] provided an analysis on the convergence basin of the global minimum for the Gauss-Newton method. Several key factors are concluded, for example orientation noises, and graph topologies. With the assumption of isometric additive Gaussian noise, Khosoussi et al. [102] established the connection between the Fisher information matrix of the estimate and the graph complexity.

Convex relaxation is a powerful technique to design globally optimal solutions to PGO. An early touch on this topic came from Liu et al. [110], who relaxed planar PGO into a semi-definite programming (SDP) which was solved by standard convex optimization tools [29]. It can be observed that the non-convexity of PGO comes from the cost function, and manifold constraints. Carlone and Rosen et al. showed that the cost function can be chosen convex by using a Frobenius norm with an isotropic noise model [35, 40, 42, 138, 140]. The manifold constraints can be relaxed by their convex-hulls, as shown by Rosen et al. [138]. However, the relaxations in [110, 138] are not tight enough. Carlone et al. [40, 42] explored the Lagrangian duality of PGO, leading to a tight SDP relaxation, which can be verified to be globally optimal in many cases. Rosen et al. [136, 140] designed a certifiable PGO solver, exploiting convex relaxations, and a Riemannian trust-region method. Briales et al. [30]

suggested a compact matrix formulation, with concise and efficient derivations.

Outliers

One of the biggest issues in terms of the robustness of GO is the characterization of spurious measurement inputs, which are often termed "outliers". A classical method to deal with outliers in the least squares community is to use the so-called "M-estimators" [88, 89, 158]. The idea is to use a robust cost function to suppress the impact of the measurements with large residuals (i.e., a potential outlier), which turns out can be equivalently solved as a reweighted least squares optimization. This method has been implemented in the mainstream GO libraries, and has served as a benchmark algorithm to other robust techniques. We refer interested readers to [135] for more detailed exposition on this technique; while we briefly mention relevant works for outlier detection in SLAM.

Sunderhauf et. al [148] introduced switchable variables into GO, aiming to downweigh the impact of outliers in the minimization process. Lee et. al [108] introduced a control variable in PGM and adopted an expectation-maximization framework to iteratively estimate its value. Both these works are shown to be closely related to reweighted least squares optimization [5, 108], and equivalent to M-estimators. An outlier in a probabilistic sense can be interpreted as a datum extremely inconsistent with its measurement model. Olson et. al [125] proposed to use a max-mixture of Gaussian to accommodate multi-modal noise scenarios. Graham et. al [74] suggested to adjust the covariance matrix for each measurement to suppress the impact of outliers. Latif et. al [106] and Graham et. al [75] used the optimal value of the cost function as a metric to find a consistent set of measurements immune from outliers. However, to obtain the optimal value, the authors resulted to brutally solving many subproblems which is rather expensive in practice. Olson et. al [126] visualized loop-closing hypotheses in SLAM as a graph, and developed a spectral graph partitioning algorithm to identify a maximal consistent set. Carlone et. al [39] proposed a mathematical formalism for outlier detection based on a ℓ_0 optimization, which was solved by a ℓ_1 relaxation followed by

a linear programming. Pfingsthorn et. al [131] bridged the gap between the SLAM front-end and back-end, by generalizing multiple loop-closing hypotheses as a hyper-edge. Then a so-called "prefilter method" is used to reduce the hyper-graph to a standard GO which will be further optimized with robust back-end techniques. Carlone et. al [33] proposed a convex relaxation for PGO with outliers.

Cycles and Metrics

As a major application of GO, SLAM has a rich history in using cycles to characterize graph topology as well. In SLAM, a cycle in the graph corresponds to the concept of "loop-closure", and we will use these two concepts interchangeably when talking about SLAM. Estrada et al. [64] formulated the loop-closing problem between local maps as a quadratic optimization problem with equality constraints. The problem was solved by SQP, and a connection to iterated extended Kalman filter was drawn. Olson [128] evaluated the pairwise consistency of two loop-closing edges by joining them with the odometry sequence as a cycle. Dubbelman et al. [59] employed interpolations in SE(3) along a cycle to obtain an approximate solution to pose-chain SLAM. The concept of cycle bases was used by Carlone et al. in [32, 34, 38]. The loop-closing cycle in a point-feature based SLAM was considered by Bai et al. [14], while both point and line-features are included by Guo et al. [80] in more specific scenarios. Later, Bai et al. [15] formulated PGO explicitly as a constrained optimization problem by using cycles in the graph.

In many robotic tasks, we rely on metrics to make decisions. The most widely used metrics in robotics, at least in estimation tasks, are attributed to the information theoretic metrics [1]. This is a broad class, and we mention several of them for completeness: (1) Fisher Information Matrix (FIM). If the processing noise is Gaussian, the metric always boils down to the evaluation of a proper covariance matrix with respect to its eigen value, trace or determinant [2][3]. (2) Mutual Information. This metric is quite popular which has been extensively used in robotics, for example, the active control/planning strategy [4][5][6][7][8],

graph pruning [9][10], online sparse pose-graph construction [11][12], path planning on a pose graph [13] and so forth. (3) Kullback-Leibler Divergence (KL Divergence). While the first two metrics deal merely with uncertainty, KL divergence can incorporate the mean and covariance of two Gaussian distributions together. This is quite similar to the optimal value in the optimization context that we will discuss later, but they are obviously in different tracks. See [14] for an example of graph pruning using KL divergence.

Chapter Three

Cycle Based Pose Graph Optimization

Pose graph optimization (PGO) is a fundamental problem which arises in various research disciplines, like simultaneous localization and mapping (SLAM) [31, 57, 122, 123, 144], structure from motion [13, 83, 151], calibration of multi-camera rig [63], and sensor network localization [130, 153]. A pose graph is a graph whose vertices encode positions and orientations of 3D poses, and whose edges represent spatial constraints between the connected vertices. Taking a graph-based SLAM system as an example, the system processes the raw measurements to construct local maps. These local maps are then arranged in a pose graph as vertices. Constraints between local maps arise from matching nearby local maps or from proprioceptive measurements coming from odometry or inertial measurement units (IMU).

In real SLAM applications, the number of edges is typically proportional to the number of vertices. This is a consequence of the local nature of SLAM, stemming from the limits in the sensor range. Only local maps that are spatially close can share some common elements, and thus the corresponding vertices can be connected by constraints (i.e., edges). This results in limiting the number of edges connected to a vertex, and ultimately leads to a sparsely connected graph. By leveraging on this sparsity, modern PGO systems [52, 92, 105, 140] are capable of solving extremely large problems in a fraction of seconds. We term the method in [52, 92, 105, 140] as vertex-based approaches for a reason that will explain later on.

At its core, the state-of-the-art PGO techniques solve a sparse linear system to update

the estimates of vertices [52, 92, 105, 140]. This system is typically solved by a sparse Cholesky factorization [48], which is guided by the graph topology presented as a vertex-edge incidence matrix. In graph theory, the incidence matrix spans a space called cut space, which is orthogonal complementary to a space called cycle space. A sparse graph implies two facts: (a) low connectivity between vertices, which has been reflected in the incidence matrix and exploited effectively [48]; (b) low dimensionality of cycle space, which has been largely ignored due to the huge success of sparse Cholesky factorization with respect to vertices. In this chapter, we will show the possibility of designing effective PGO techniques in the graph cycle space.

In this chapter, we will formulate and solve PGO in the cycle space, and provide insights into the observability and convergence of this novel PGO method. In line with the common sense that the reduction of the dimension would increase the density of the problem (so that no information is lost during the process), we show how to maximize the sparsity of the cycle based formulation by using a minimum cycle basis (MCB). While MCB is considered expensive, we bring together the recent insights in graph theory, and design a MCB algorithm that is comparable (actually slightly faster) to the existing state-of-the-art. Numerical experiments are provided to show the superior convergence property of the cycle based PGO (by using a MCB), in terms of both the convergence rate and convergence to the global minimum. To foster the reproduction of the results, we provide a complete open-source C++ implementation[1] of our approach.

The notions are in accordance with the stipulation in Section 2.1. In addition, there is an intensive exposition to graph concepts in this chapter, particularly for the description of the minimum cycle basis. For easy reference, the graph notations are listed in Table 3.1.

[1]Code repository: https://bitbucket.org/FangBai/slam_pgo_cycle_vs_vertex/src/master/

Table 3.1 List of Notations on Graph

$\mathbb{GF} = \mathbb{Z}_2$		finite field of modulo 2
$\mathbb{GF}^{d_1 \times d_2}$		$d_1 \times d_2$ dimensional matrix on $\mathbb{Z}_2$
$\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$		undirected graph
$\mathcal{E}$		edge set of a graph $\mathcal{G}$
$\mathcal{V}$		vertex set of a graph $\mathcal{G}$
e_{uv}	$\mathcal{E}$	edge with endpoints u, v
$\nu = \lvert\mathcal{E}\rvert - \lvert\mathcal{V}\rvert + 1$	$\mathbb{R}$	dimension of cycle space
$\mathcal{T} = \mathcal{T}(\mathcal{G})$	$\mathbb{GF}^{1\times m}$	tree in $\mathcal{G}$
$\mathcal{P}_{uv} = \mathcal{P}_{uv}(\mathcal{G})$	$\mathbb{GF}^{1\times m}$	path in $\mathcal{G}$, from vertex u to v
$\mathcal{C} = \mathcal{C}(\mathcal{G})$	$\mathbb{GF}^{1\times m}$	cycle in $\mathcal{G}$
$\mathcal{S}$	$\mathbb{GF}^{1\times m}$	support vector
Note: A vector on $\mathbb{GF}^{1\times m}$ can be sparsely described by a set.		
$\{*\}$		set with elements in form of $*$
$\boldsymbol{\mathcal{H}}$	$\{\mathcal{C}\}$	Horton set
$\boldsymbol{\mathcal{I}}$	$\{\mathcal{C}\}$	set of isometric cycles
$\boldsymbol{\mathcal{B}}$	$\{\mathcal{C}\}$	cycle basis
$\mathcal{B} = \mathcal{B}(\mathcal{G})$	$\mathbb{GF}^{n\times\nu}$	cycle matrix of cycle basis $\boldsymbol{\mathcal{B}}$
$\mathbf{T}_k$	SE(3)	poses
$\mathbf{T_k}$	SE(3)	relative poses
$\boldsymbol{\mathcal{P}}_{uv}$	$\{\mathbf{T_k}\}$	geometric path from u to v
$\boldsymbol{\mathcal{C}}$	$\{\mathbf{T_k}\}$	geometric cycle

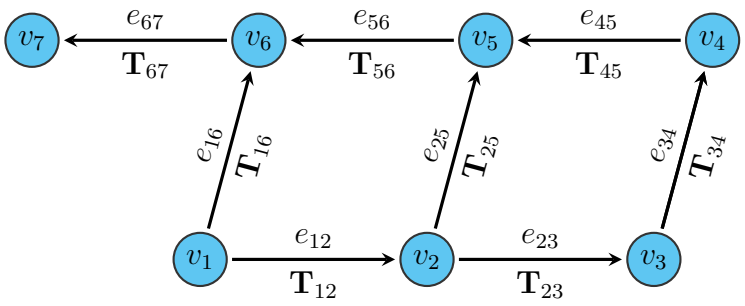


Figure 3.1 A toy graph of PGO. For each relative poses $\mathbf{T}_{i,j}$, the edge e_{ij} is oriented as $i \rightarrowtail j$. When talking about topological information, like cycles/paths, we can safely operate on the undirected version by ignoring the edge orientations, and lifting back to oriented edges when the cycles/paths are computed.

3.1 Traditional Pose-Graph Optimization

A pose graph optimization (PGO) is a special case of graph optimization, where the unobserved latent variable is called poses, and the observed variable is called relative poses. In PGO, both poses and relative poses are rigid-body transformations, which can be described by SE(3) Lie group. Denote $\mathbf{T}_i$ to be the i-th pose. The relative poses from the i-th pose to the j-th pose takes the form $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j$. In a factor graph representation, the factor node $\mathbf{T}_{i,j}$ is always binary, with exactly two vertices i and j connecting to it. Therefore, a more intuitive graph representation of PGO is to encode the factors with edges explicitly, that the edge e_{ij} corresponds to the relative poses $\mathbf{T}_{i,j}$. Fig. 3.1 is a toy example of pose graph described explicitly with the vertex-edge paradigm.

The topology of PGO can be visualized as a graph whose vertices represent poses. An edge is created if there exists a relative geometric relation between two poses, i.e. relative poses, which can be produced by a wheel-encoder, scan-matching [24, 113, 146], epipolar geometry [82, 142], or visual loop-closing techniques etc. [47, 112]. Both poses and relative poses are rigid-body transformations, which can be described by SE(3) Lie group. Denote $\mathbf{T}_i$ to be the i-th pose. The relative poses from the i-th pose to the j-th pose, denoted by $\mathbf{T}_{i,j}$, is a rigid-body transformation evaluated in the local coordinate frame of the i-th pose, which mathematically writes $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j$. For the clarity of notations, we assign each edge

a unique index, and use $\mathbf{T_k}$ with subscript in bold to represent a relative poses.

For each relative pose $\mathbf{T_k}$, there is a noisy measurement $\tilde{\mathbf{T}}_\mathbf{k}$. The measurement noise is conventionally assumed to be zero-mean Gaussian in the vector space of SE(3) Lie algebra, which can be mathematically formalized as,

$$\mathbf{Log}(\tilde{\mathbf{T}}_\mathbf{k}^{-1} \cdot \mathbf{T_k}) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_k}).$$

Note that other noise models are also possible, for example, the matrix Langevin distributions used in [42, 140].

Let the topology of PGO be described by the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Then PGO aims to obtain a maximum likelihood (MLE) estimator for the set of poses $\{\mathbf{T}_i\}_{i\in\mathcal{V}}$ using the set of measurements of relative poses $\{\tilde{\mathbf{T}}_\mathbf{k}\}_{\mathbf{k}\in\mathcal{E}}$, via solving a least squares optimization problem,

$$\{\mathbf{T}_i\}_{i\in\mathcal{V}} = \operatorname{argmin} \sum_{\mathbf{k}\in\mathcal{E}} \|\mathbf{Log}(\tilde{\mathbf{T}}_\mathbf{k}^{-1} \cdot \mathbf{T_k})\|^2_{\mathbf{\Sigma_k}}. \tag{3.1}$$

Remark 1. Note that the relative poses $\mathbf{T}_{i,j}$ is evaluated at the local frame of the i-th pose, so ideally a PGO is described by a directed graph. However, the edge orientation will not affect the topological side of a graph, like paths/cycles we discuss later on. Therefore we opt to describe PGO as an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The restriction to the undirected graph limits the graph matrices/vectors to Galois field instead of real numbers. Besides, the undirected edges can be easily lifted to directed edges whenever it is desired.

Remark 2. The traditional PGO is unobservable, in the sense that if $\{\mathbf{T}_i\}_{i\in\mathcal{V}}$ is a solution to (3.1), then $\forall \mathbf{T}' \in \mathrm{SE}(3)$, $\{\mathbf{T}'\mathbf{T}_i\}_{i\in\mathcal{V}}$ is also a solution to (3.1). This can be easily verified by the fact that $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j = (\mathbf{T}'\mathbf{T}_i)^{-1}(\mathbf{T}'\mathbf{T}_j)$, thus $\{\mathbf{T}_i\}_{i\in\mathcal{V}}$ and $\{\mathbf{T}'\mathbf{T}_i\}_{i\in\mathcal{V}}$ yields exactly the same contribution in the cost function. To ensure a unique solution, a popular practice is to anchor some poses (usually the first pose) to a fixed value or the identity of SE(3).

3.2 Cycle Based Pose-Graph Optimization

The state-of-the-art PGO techniques [42, 52, 92, 105, 139] describe PGO as a factor graph [54], whose topology is represented by an incidence matrix. The graph is solved by a second-order optimization technique, for example Gauss-Newton, which results to solve a sparse linear system whose dimension is decided by the number of vertices. These approaches solve PGO in the cut space, and we will term them as vertex-based PGO (VB-PGO). The VB-PGO can be solved rather efficiently by sparse matrix factorizations [48].

Practical PGO instances are rather sparse. Let us measure the graph sparsity as a concept called cycle ratio, defined as $\frac{\nu}{|\mathcal{E}|}$, i.e., the dimension of the cycle space divided by the number of edges. Empirically, a PGO instance encountered in SLAM has a cycle ratio well below 20%, which implies $\frac{\nu}{|\mathcal{V}|-1} < 1/4$, i.e., the dimension of the cut space is at least four times larger than that of the cycle space. Let alone SLAM instances with a long trajectory and a few loop-closures, whose cycle ratio can be less than 5%, or even 1%.

In this section, we will describe an approach that transforms PGO from the cut space to the cycle space. The cycle-based PGO, denoted as CB-PGO, has a reduced dimension compared with its vertex-based counterpart, as long as the graph is sparse enough. While the dimension reduction to the cycle space can undermine the sparsity of PGO, we propose to maximize the sparsity by a minimum cycle basis which will be described in Section 3.3. The observability and convergence properties of the proposed CB-PGO are discussed in Section 3.5.

3.2.1 Topological and Geometric Paths/Cycles

In what follows, we will use the term toplogical path and toplogical cycle to represent a path and cycle in a pure topological graph $\mathcal{G}$. If such a $\mathcal{G}$ is associated with geometric information, namely by associating vertices with poses and edges with relative poses respectively, we will term this graph as a geometric graph. A (topological) path $\mathcal{P}$ whose edges are associated with

relative poses will be termed as a geometric path, denoted by $\boldsymbol{\mathcal{P}}$. Analogously, a (topological) cycle $\mathcal{C}$ with edges associated with relative poses will be termed as a geometric cycle, denoted by $\boldsymbol{\mathcal{C}}$. If not explicitly stated, the terms, paths and cycles, refer to the topological version.

The orientations of edges are irrelevant in this paper when discussing the topology of $\mathcal{G}$, as well as concepts like cycle bases and sparsity. However, they are useful in terms of describing the geometric paths/cycles. The orientation of an edge is decided by the geometric information, i.e., relative poses it associated with. For example, given an edge $\mathbf{k}$ in $\mathcal{G}$ with the associated relative poses being $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j$, we stipulate the edge orientation to be from i to j. In other words, $i \rightarrowtail j$ is the forward direction of the edge $\mathbf{k}$, and $j \rightarrowtail i$ is the backward direction.

3.2.2 Consistency of Pose-Graph Optimization

We will use the term "consistency" to express the point that all the noise-free measurements encoded on edges of the graph must be consistent with the real pose position of the robot. The consistency says: All noise-free measurements are decided by the unique pose configuration in the real environment, so the noise-free measurement readings must not contradict one other. The consistency property in the noise-free case must be satisfied for the real robot trajectory, as well as for the the optimal estimate.

Let $\mathcal{P}_{st}$ be a path from s to t in $\mathcal{G}$. The corresponding geometric path $\boldsymbol{\mathcal{P}}_{st}$ is defined as:

$$\boldsymbol{\mathcal{P}}_{st} = \mathbf{T}_{\mathbf{1}^p}^{\varrho(\mathbf{1}^p)}\mathbf{T}_{\mathbf{2}^p}^{\varrho(\mathbf{2}^p)}\cdots\mathbf{T}_{\boldsymbol{\theta}^p}^{\varrho(\boldsymbol{\theta}^p)} \tag{3.2}$$

where $\mathbf{T}_{\mathbf{1}^p}, \mathbf{T}_{\mathbf{2}^p}, \cdots, \mathbf{T}_{\boldsymbol{\theta}^p}$ is the sequence of relative poses by traversing $\mathcal{P}_{st}$ from s to t. In (3.2), the superscript $\varrho(\mathbf{k}^p)$ is assigned to $+1$ if the traversal uses the edge $\mathbf{k}^p$ in the forward direction, and -1 if in the backward direction. The superscript of a matrix $\mathbf{T}$ will be interpreted as the power of the matrix, by $\mathbf{T}^{+1} = \mathbf{T}$ and $\mathbf{T}^{-1} = \mathbf{inv}(\mathbf{T})$.

PGO is a consistent formulation with respect to poses and relative poses. To show this, let $\mathbf{T}_s$ and $\mathbf{T}_t$ be two arbitrary poses. Let $\boldsymbol{\mathcal{P}}_{1:st}$ and $\boldsymbol{\mathcal{P}}_{2:st}$ be two geometric paths from

pose s to t. Then the pose $\mathbf{T}_t$ calculated from these two paths are exactly the same, i.e., $\mathbf{T}_t = \mathbf{T}_s\boldsymbol{\mathcal{P}}_{1:st} = \mathbf{T}_s\boldsymbol{\mathcal{P}}_{2:st}$. Obviously, the consistency of the PGO can be also interpreted as the equivalence of the geometric paths, in the sense that $\boldsymbol{\mathcal{P}}_{1:st} = \boldsymbol{\mathcal{P}}_{2:st} = \mathbf{T}_s^{-1}\mathbf{T}_t$. At last, with a slightly abuse of notation, we can write the consistency of two geometric paths as $\boldsymbol{\mathcal{P}}_{1:st}^{-1}\boldsymbol{\mathcal{P}}_{2:st} = \mathbf{I}$, which is a geometric cycle.

The geometric cycles will play a key role in formulating PGO in cycle space, which in general ensures the consistency of the PGO. On the other hand, the underlying topological cycles will decide the sparsity of the proposed PGO formulation.

3.2.3 PGO in Cycle Space

Alternatively, we can traverse edges sequentially along a topological cycle, and consider the associated relative poses, to obtain a geometric cycle. For example, consider a topological cycle with length $\boldsymbol{\lambda}$. Let the sequential relative poses along the cycle be $\mathbf{T}_{\mathbf{1}^c}, \mathbf{T}_{\mathbf{2}^c}, \cdots, \mathbf{T}_{\boldsymbol{\lambda}^c}$, and the corresponding orientations of the edges be $\sigma(\mathbf{1}^c), \sigma(\mathbf{2}^c), \cdots, \sigma(\boldsymbol{\lambda}^c)$, where $\sigma(\mathbf{k}^c)$ takes value $+1$ if the traversal uses the edge $\mathbf{k}^c$ in the forward direction, and -1 otherwise. Then the corresponding geometric cycle can be written as follow,

$$\boldsymbol{\mathcal{C}}^{\text{lhs}} = \mathbf{I}, \quad \text{with} \;\; \boldsymbol{\mathcal{C}}^{\text{lhs}} = \mathbf{T}_{\mathbf{1}^c}^{\sigma(\mathbf{1}^c)}\mathbf{T}_{\mathbf{2}^c}^{\sigma(\mathbf{2}^c)}\cdots\mathbf{T}_{\boldsymbol{\lambda}^c}^{\sigma(\boldsymbol{\lambda}^c)}$$

where $\mathbf{T}^{+1} = \mathbf{T}$ and $\mathbf{T}^{-1} = \mathbf{inv}(\mathbf{T})$ respectively.

Based on the edge orientations and associated relative poses, for each topological cycle $\mathcal{C}$, we can generate a corresponding geometric cycle $\boldsymbol{\mathcal{C}}^{\text{lhs}} = \mathbf{I}$. To characterize the cycle space of the graph $\mathcal{G}$, we need ν independent topological cycles, i.e., a cycle basis of $\mathcal{G}$. Let such a cycle basis be $\boldsymbol{\mathcal{B}} = \{\mathcal{C}_i\}_{i=[1:\nu]}$, and its corresponding cycle matrix be $\mathcal{B}$. Then given the cycle basis $\boldsymbol{\mathcal{B}}$, we can find ν independent geometric cycles accordingly, denoted as $\{\boldsymbol{\mathcal{C}}_i^{\text{lhs}} = \mathbf{I}\}_{i=[1:\nu]}$.

Then let us take all $|\mathcal{E}|$ relative poses as new variables to be estimated, and take ν

independent geometric cycles as constraints in an optimization problem:

$$\begin{aligned} \{\mathbf{T_k}\}_{\mathbf{k}\in\mathcal{E}} = \text{argmin } \textstyle\sum_{\mathbf{k}\in\mathcal{E}} \|\mathbf{Log}(\tilde{\mathbf{T}}_{\mathbf{k}}^{-1}\cdot\mathbf{T_k})\|^2_{\mathbf{\Sigma_k}} \\ \mathbf{s.t.} \quad \boldsymbol{\mathcal{C}}_i^{\text{lhs}} = \mathbf{I} \quad \forall i \in [1:\nu]. \end{aligned} \tag{3.3}$$

This optimization problem has a degree-of-freedom (DOF) $|\mathcal{E}| - \nu = |\mathcal{V}| - 1$, which is the same as the DOF of (3.1). If an optimal configuration of relative poses is found by solving (3.3), the objective value becomes minimum and all paths between two vertices in the graph become equivalent (guaranteed by the geometric constraints). Then a solution to (3.1) can be calculated by composing the estimates of relative poses along an arbitrary path in the graph (for example along odometry).

In what follows, we will term the PGO formulation in (3.3) as cycle-based PGO (**CB-PGO**), and in contrast, the PGO formulation in (3.1) as vertex-based PGO (**VB-PGO**).

3.2.4 Solving Cycle Based PGO on Manifold

A typical iterative optimization algorithm on Manifold is driven by a sequence of small perturbations until convergence (to a local optimum). For SE(3), the perturbations are normally applied in the vector space of its Lie algebra, which can be passed to the manifold via the exponential mapping. To solve the PGO formulation in (3.3), at each iteration t, we would like to find a perturbation $\boldsymbol{\xi}_\mathbf{k}$ for each relative poses $\mathbf{T_k}$, so that its estimate can evolve from $\hat{\mathbf{T}}_\mathbf{k}^{(t)}$ (estimate at iteration t) to $\hat{\mathbf{T}}_\mathbf{k}^{(t+1)}$ (estimate at iteration $t+1$) as,

$$\hat{\mathbf{T}}_\mathbf{k}^{(t+1)} = \hat{\mathbf{T}}_\mathbf{k}^{(t)} \cdot \mathbf{Exp}\left(\boldsymbol{\xi}_\mathbf{k}\right), \quad \mathbf{k}\in\mathcal{E}.$$

To this end, we linearize the PGO formulation in (3.3) with respect to the set of perturbations, to a quadratic programming with equality constraints,

$$\min \|\mathbf{J}^{-1}\boldsymbol{\xi} + \boldsymbol{\eta}\|^2_{\mathbf{\Sigma}} \qquad \mathbf{s.t.} \quad \mathbf{B}\boldsymbol{\xi} + \mathbf{b} = \mathbf{0}. \tag{3.4}$$

Here $\mathbf{J}$ and $\boldsymbol{\eta}$ are the Jacobian matrix and the residual vector respectively by linearizing the objective function, whose calculations are provided in Appendix A.1. Analogously, $\mathbf{B}$ and $\mathbf{b}$

are the Jacobian matrix, and its corresponding residual vector by linearizing the geometric cycles. The details on how to derive $\mathbf{B}$ and $\mathbf{b}$ can be found in Appendix A.2. Note that the i-th block row and $\mathbf{k}$-th block column of $\mathbf{B}$ represents the partial derivative of the i-th geometric cycle with respect to the $\mathbf{k}$-th edge (i.e., relative poses), which is non-zero iff the $\mathbf{k}$-th edge is contained in the i-th cycle. In other words, the structure of $\mathbf{B}$ is captured by the cycle matrix $\mathcal{B}$.

By letting $\bar{\boldsymbol{\xi}} = \boldsymbol{\Sigma}^{-\frac{1}{2}}(\boldsymbol{\eta}+\mathbf{J}^{-1}\boldsymbol{\xi})$, $\bar{\mathbf{B}} = \mathbf{B}\mathbf{J}\boldsymbol{\Sigma}^{\frac{1}{2}}$, and $\bar{\mathbf{b}} = \mathbf{B}\mathbf{J}\boldsymbol{\eta}-\mathbf{b}$, the quadratic programming in (3.4) takes the form of a minimum norm optimization problem,

$$\min \|\bar{\boldsymbol{\xi}}\|^2 \qquad \mathbf{s.t.} \quad \bar{\mathbf{B}}\bar{\boldsymbol{\xi}} = \bar{\mathbf{b}} \tag{3.5}$$

whose solution is $\bar{\boldsymbol{\xi}}^{\text{opt}} = \bar{\mathbf{B}}^{\dagger}\bar{\mathbf{b}}$. Note that since both $\boldsymbol{\Sigma}^{\frac{1}{2}}$ and $\mathbf{J}$ are block-diagonal matrices, $\bar{\mathbf{B}}$ and $\mathbf{B}$ would have the same structure. Finally, the perturbation in $\boldsymbol{\xi}$ can be recovered by $\boldsymbol{\xi}^{\text{opt}} = \mathbf{J}(\boldsymbol{\Sigma}^{\frac{1}{2}}\bar{\boldsymbol{\xi}}^{\text{opt}} - \boldsymbol{\eta})$.

The overall algorithm can be termed as sequential quadratic programming (SQP) [14, 15, 26], since it requires the solving of a sequence of perturbations via quadratic programming.

Remark 3. The linear system $\bar{\boldsymbol{\xi}}^{\text{opt}} = \bar{\mathbf{B}}^{\dagger}\bar{\mathbf{b}}$ to be solved in CB-PGO has exactly the same dimension of that in the cycle space, which is $\nu = |\mathcal{E}| - |\mathcal{V}| + 1$, because the Jacobian is characterized by the cycle matrix $\mathcal{B}$. In contrast, an iterative solver to VB-PGO in (3.1) solves a linear system with a dimension of $|\mathcal{V}| - 1$, i.e, the dimension of cut space, since its Jacobian is characterized by the incidence matrix [32, 38, 101]. Given the fact that the cut space and cycle space are orthogonal complementary [134], we conclude from the graph topology perspective that the VB-PGO and CB-PGO are counterparts to one another. Moreover, VB-PGO is a least squares optimization for an over-determinant system, while CB-PGO is a minimum norm optimization for an under-determinant system. Mathematically, the least squares optimization and minimum norm optimization are actually the same thing [120], where both solutions are compactly written as Moore-Penrose pseudo inverse (i.e., left and right inverse respectively). This fact further confirms that VB-PGO in (3.1) and CB-

PGO (3.3) are two sides of the same coin. However, while VB-PGO has reached a mature state, the CB-PGO technique is still rather primitive, because of the hardship of choosing a proper cycle basis.

3.2.5 Choices of Cycle Basis for PGO

For the cycle-based PGO in (3.3), the structure of the Jacobian matrices $\mathbf{B}$ and $\bar{\mathbf{B}}$ is completely described by a cycle matrix $\mathcal{B}$. Obviously, different choices of cycle bases $\boldsymbol{\mathcal{B}}$ lead to different cycle matrices $\mathcal{B}$, which eventually lead to different structures in $\mathbf{B}$ and $\bar{\mathbf{B}}$.

Therefore, we discuss here the pros and cons of different cycle bases for the PGO formulation in (3.3). We will conclude the advantage of using a minimum cycle basis (MCB) from the sparsity perspective. The discussions on the convergence behavior will be presented in Section 3.5.2.

Fundamental Cycle Basis

Given an arbitrary spanning tree of the graph, a cycle can be constructed by one off-tree edge (i.e. chord), and the path on the tree connecting the ends of the edge. The set of cycles corresponding to these ν off-tree edges are independent, called fundamental cycle basis (FCB). FCB can be generated cheaply, while it cannot ensure a sparse Jacobian matrix in general [15].

Minimum Fundamental Cycle Basis

A remedy is to use the minimum fundamental cycle basis (MFCB), where the spanning tree is chosen in a way such that the summation of the lengths of the fundamental cycles is minimum. An exact solution to MFCB is proven to be NP-complete [55]. While approximate algorithms can solve MFCB in polynomial time [9, 55, 69], constraining cycle basis to be fundamental may compromise the sparsity.

Minimum Overlap Cycle Basis

In light of the fact that the matrix to be factorized has the same structure as $\mathcal{B}\mathcal{B}^T$, the sparseness of the matrix decomposition can be guaranteed by minimizing the number of non-zeros in $\mathcal{B}\mathcal{B}^T$. We name a cycle basis that minimizes $|\mathcal{B}\mathcal{B}^T|_0$ as the minimum overlap cycle basis (MOCB), by the fact that an entry at position (i, j) in $|\mathcal{B}\mathcal{B}^T|_0$ is 0 if and only if the cycle i and j do not share common edges. However, there is no clear way on how to compute a minimum overlap cycle basis yet.

Minimum Length Cycle Basis

A minimum length cycle basis (MLCB) maximizes the sparsity of $\mathcal{B}$, by minimizing the overall length of cycles in the basis, which may in turn resulting a sparse $\mathcal{B}\mathcal{B}^T$. Different from MFCB, the cycles are not confined to be fundamental, thus yielding an easier problem. MLCB belongs to a well-known problem termed minimum (weight) cycle basis (MCB) [100], where MLCB is a special case with weights of edges set to 1.

According to the discussion above, we opt to use MLCB for the cycle-based PGO to maximize the sparsity. Since MLCB is a special case of the general MCB, we focus on how to compute MCB in the following Section 3.3. In what follows, we will use the term MCB only, while the same algorithm can be used to compute MLCB trivially by assigning the weights of all edges uniformly, for example to 1.

3.3 Computing Minimum Cycle Basis

In this section, we aim at a complete and concise description of the MCB algorithm used for the cycle-based PGO. We will follow a hybrid approach that firstly construct a superset that contains MCB, and then apply an independence test to extract a MCB. We will recall the basics of these concepts for completeness, in particular the construction of Horton set

Algorithm 1 Minimum Cycle Basis

$\bar{\mathcal{G}} \leftarrow$ SimplifyGraph ($\mathcal{G}$) ▷ Smooth out vertices of degree two

APSP $\leftarrow$ LexDijkstra ($\bar{\mathcal{G}}$) ▷ Compute a set of consistent all-pairs-shortest-paths

$\boldsymbol{\mathcal{H}} \leftarrow$ HortonSet (APSP) ▷ Construct Horton set implicitly

$\boldsymbol{\mathcal{I}} \leftarrow$ IsometricSet ($\boldsymbol{\mathcal{H}}$) ▷ Construct isometric set

$\bar{\boldsymbol{\mathcal{B}}} \leftarrow \emptyset$ ▷ Initialize MCB for $\bar{\mathcal{G}}$

$\boldsymbol{\mathcal{I}} \leftarrow$ SortAscendingByWeight ($\boldsymbol{\mathcal{I}}$)

// Independence test by support vectors

while $|\bar{\boldsymbol{\mathcal{B}}}| \neq \nu$ **do**

  $\mathcal{C} \leftarrow$ ExtractMinimumWeightCircuit ($\boldsymbol{\mathcal{I}}$)

  if $\mathcal{C}$ is linearly independent from $\bar{\boldsymbol{\mathcal{B}}}$ **then**

    $\bar{\boldsymbol{\mathcal{B}}} \leftarrow \bar{\boldsymbol{\mathcal{B}}} \cup \mathcal{C}$

  end if

  $\boldsymbol{\mathcal{I}} \leftarrow \boldsymbol{\mathcal{I}} \backslash \mathcal{C}$

end while

$\boldsymbol{\mathcal{B}} \leftarrow$ ReconstructMCB ($\bar{\boldsymbol{\mathcal{B}}}$) ▷ Reconstruct MCB for $\mathcal{G}$

[85] and isometric set [8], and the state-of-the-art method for independence tests [10], while refer interested reader to the review paper [100] for further reading.

On the present architecture, the computational bottleneck of MCB algorithms is the all-pairs-shortest-paths (APSP) [62, 119]. APSP is required to construct the Horton set, and a consistent APSP for the isometric set. Given the fact that graphs occurred in PGO are sparse graphs with positive integer weights, we propose two ideas that can substantially improve the performance of APSP: 1) smoothing out vertices of degree two; 2) using LexDijkstra (in Section 3.3.6) to compute a consistent APSP that can run in parallel, and thus is more advantageous than the sequential method in [84].

The overall procedure of the proposed MCB algorithm is summarized in Algorithm 1.

3.3.1 Superset of MCB: Horton Set

The study of efficient polynomial time minimum cycle basis algorithms started with Horton's work [85], which builds the connection of shortest paths and a cycle in MCB.

Lemma 1. ([85]). Let $\mathcal{C}$ be a cycle in a minimum cycle basis $\boldsymbol{\mathcal{B}}$. If u and v are two vertices on $\mathcal{C}$, then $\mathcal{C}$ must contain one of the shortest paths from u to v.

Another key insight from Horton [85] is that using shortest paths, each cycle $\mathcal{C} \in \boldsymbol{\mathcal{B}}$ can be represented as a vertex-edge pair, called a representation of a cycle. In specific, let x be any vertex in $\mathcal{C}$ ($\mathcal{C} \in \boldsymbol{\mathcal{B}}$), then we can always find an edge e_{uv} such that $\mathcal{C}$ can be expressed as $\mathcal{C} = \mathcal{C}(x, e_{uv}) \triangleq \mathcal{P}_{xu} + \mathcal{P}_{xv} + e_{uv}$, where $\mathcal{P}_{xu}$ and $\mathcal{P}_{xv}$ are shortest paths. Based on this observation, Horton [85] proposed a superset (called Horton set) of MCB using all pairs of vertex-edge combinations. Formally, a Horton set is defined as,

$$\boldsymbol{\mathcal{H}} = \{\mathcal{C}(x, e_{uv}) \mid x \in \mathcal{V},\ e_{uv} \in \mathcal{E}\}.$$

Note that there might be several shortest paths between the vertices u and v with exactly the same minimum weight, so the choice of $\mathcal{P}_{uv}$ is not unique in general. Horton [85] proved that if all edges in the graph have nonnegative weights, $\boldsymbol{\mathcal{H}}$ would definitively contain a MCB no matter what shortest path $\mathcal{P}_{uv}$ is chosen for each pair of vertices u and v.

Remark 4. Typically, $\boldsymbol{\mathcal{H}}$ is much larger than $\boldsymbol{\mathcal{B}}$. On the one hand, there are degenerated cases in $\boldsymbol{\mathcal{H}}$ that do not form a simple cycle, for example, if e_{uv} is on $\mathcal{P}_{xu}$ or $\mathcal{P}_{xv}$, or if $\mathcal{P}_{xu}$ and $\mathcal{P}_{xv}$ have vertices other than x in common. However the degenerated cases can be easily removed. On the other hand, $\boldsymbol{\mathcal{H}}$ is a multi-set. By choosing different vertex-edge pairs on $\mathcal{C}$, we obtain different representations of $\mathcal{C}$.

3.3.2 Superset of MCB: Isometric Circuits

The construction of $\boldsymbol{\mathcal{H}}$ requires the computation of all pairs shortest paths (APSP). In Horton's work [85], APSP is allowed to be arbitrary, i.e. the shortest path $\mathcal{P}_{uv}$ is selected

arbitrarily among all shortest paths from u to v. By intentionally selecting APSP to be consistent, we can identify the duplicates in $\boldsymbol{\mathcal{H}}$, and reduce $\boldsymbol{\mathcal{H}}$ to a much smaller set.

Definition 1. (Consistent APSP). For each shortest path $\mathcal{P}_{uv}$ in APSP, let s and t be two arbitrary vertices lying on $\mathcal{P}_{uv}$, then $\mathcal{P}_{st}$, the selected shortest path from s to t in APSP, is a subgraph of $\mathcal{P}_{uv}$.

A consistent APSP can be computed by a lexicographic method [84, 100].

Definition 2. (Lexicographic Paths [84]). Consider undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with edge weight vector w. Each edge in $\mathcal{E}$ is assigned with a unique index, and a path $\mathcal{P}$ is described as a collection of edges. Then for every pair of nodes $u, v \in \mathcal{V}$, there exists a unique u - v path $\mathcal{P}_{uv}$, that satisfies exactly one of the following three conditions with respect to any other u - v path $\mathcal{P}'_{uv}$:

(1) $w(\mathcal{P}_{uv}) < w(\mathcal{P}'_{uv})$,

(2) $w(\mathcal{P}_{uv}) = w(\mathcal{P}'_{uv})$ and $|\mathcal{P}_{uv}| < |\mathcal{P}'_{uv}|$,

(3) $w(\mathcal{P}_{uv}) = w(\mathcal{P}'_{uv})$, $|\mathcal{P}_{uv}| = |\mathcal{P}'_{uv}|$, and $\text{min\_index}(\mathcal{P}_{uv} \backslash \mathcal{P}'_{uv}) < \text{min\_index}(\mathcal{P}'_{uv} \backslash \mathcal{P}_{uv})$.

Lemma 2. ([84]) If all paths in a APSP are lexicographic paths, then the APSP is a consistent APSP.

Proof. The proof is not given in [84], thus we provide one for completeness. Let s and t be two vertices on the lexicographic shortest path $\mathcal{P}_{uv}$. Suppose the lexicographic shortest path from s to t, denoted by $\mathcal{P}_{st}$, is not on $\mathcal{P}_{uv}$. Let $\mathcal{P}^*_{st}$ be the subgraph on $\mathcal{P}_{uv}$ joining s and t, i.e., $\mathcal{P}_{uv} = \mathcal{P}_{us} + \mathcal{P}^*_{st} + \mathcal{P}_{tv}$. Let us further define a path $\mathcal{P}'_{uv} = \mathcal{P}_{us} + \mathcal{P}_{st} + \mathcal{P}_{tv}$. Obviously, $w(\mathcal{P}^*_{st}) = w(\mathcal{P}_{st})$ and $|\mathcal{P}^*_{st}| = |\mathcal{P}_{st}|$ (which implies $w(\mathcal{P}_{uv}) = w(\mathcal{P}'_{uv})$ and $|\mathcal{P}_{uv}| = |\mathcal{P}'_{uv}|$), otherwise either $\mathcal{P}_{uv}$ or $\mathcal{P}_{st}$ is not a lexicographic shortest path by the case (1) or case (2) of Definition 2.

For the case (3) of Definition 2, it can be verified,

$$\text{min\_index}(\mathcal{P}_{uv} \backslash \mathcal{P}'_{uv}) = \text{min\_index}(\mathcal{P}^*_{st} \backslash \mathcal{P}_{st}) \tag{3.6}$$

$$\text{min\_index}(\mathcal{P}'_{uv} \backslash \mathcal{P}_{uv}) = \text{min\_index}(\mathcal{P}_{st} \backslash \mathcal{P}^*_{st}). \tag{3.7}$$

Since $\mathcal{P}_{st}$ is the lexicographic shortest path, we have

$$\text{min\_index}(\mathcal{P}_{st} \backslash \mathcal{P}^*_{st}) < \text{min\_index}(\mathcal{P}^*_{st} \backslash \mathcal{P}_{st}). \tag{3.8}$$

From Eq. (3.6)(3.7)(3.8), we conclude $\text{min\_index}(\mathcal{P}'_{uv} \backslash \mathcal{P}_{uv}) < \text{min\_index}(\mathcal{P}_{uv} \backslash \mathcal{P}'_{uv})$. Therefore by the case (3) of Definition 2, $\mathcal{P}_{uv}$ is not a lexicographic shortest path since $\mathcal{P}'_{uv}$ is lexicographically shorter than $\mathcal{P}_{uv}$, which contradicts the assumption. □

Given a consistent APSP, a cycle $\mathcal{C}$ is said to be isometric if for any two vertices u and v on $\mathcal{C}$, the chosen shortest path $\mathcal{P}_{uv}$ in APSP is contained in $\mathcal{C}$ [8, 85]. It can be further verified that a cycle $\mathcal{C}$ is isometric if and only if, for each vertex $x \in \mathcal{C}$, there is a unique edge e_{uv}, such that $\mathcal{C} = \mathcal{C}(x, e_{uv})$, with $\mathcal{P}_{xu}$ and $\mathcal{P}_{xv}$ being in the consistent APSP [8]. Last but not least, the set of all isometric cycles is proved to contain a MCB [8, 100].

In a Horton set $\boldsymbol{\mathcal{H}}$ constructed by a consistent APSP, an isometric cycle $\mathcal{C} \in \boldsymbol{\mathcal{H}}$ would contain exactly $|\mathcal{C}|$ representations, i.e. $|\mathcal{C}|$ duplicates in $\boldsymbol{\mathcal{H}}$. Aiming to eliminate redundant representations, we will use the following Lemma to find all the equivalent representations in the Horton set.

Lemma 3. ([8]). Let $s_x(y)$ be the first vertex (except x) on the shortest path $\mathcal{P}_{xy}$. For any cycle $\mathcal{C} = \mathcal{C}(x, e_{uv}) \in \boldsymbol{\mathcal{H}}$, with $e_{uv} \notin \mathcal{P}_{xu}$, $e_{uv} \notin \mathcal{P}_{xv}$, and $s_x(u) \neq s_x(v)$,

(1) if $x = u$ then $\mathcal{C} = \mathcal{C}(v, e_{uv})$.

(2) if $x \neq u$, let $x' = s_x(u)$,

  (a) if $x = s_{x'}(v)$ then $\mathcal{C} = \mathcal{C}(x', e_{uv})$.

  (b) if $x \neq s_{x'}(v)$, $u = s_v(x')$ then $\mathcal{C} = \mathcal{C}(v, e_{xx'})$.

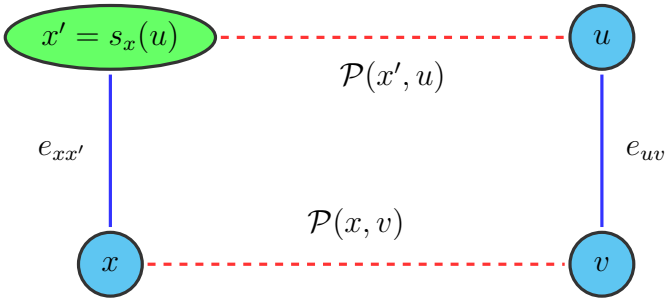


Figure 3.2 An illustration of Lemma 3. Let us consider the cycle represented by the vertex x and edge e_{uv}, i.e. $\mathcal{C} = \mathcal{C}(x, e_{uv}) \triangleq \mathcal{P}(x, u) + \mathcal{P}(x, v) + e_{uv}$. $x' = s_x(u)$ is the first vertex on the path $\mathcal{P}(x, u)$, i.e., $\mathcal{P}(x, u) = e_{xx'} + \mathcal{P}(x', u)$. The key to the proof of Lemma 3 is based on the observation that: If $\mathcal{C}$ is isometric, there are two possible cases for the shortest path between the vertex x' and v: (a) $\mathcal{P}(x', v) = e_{xx'} + \mathcal{P}(x, v)$, (b) $\mathcal{P}(x', v) = \mathcal{P}(x', u) + e_{uv}$. Obviously case (a) implies $x = s_{x'}(v)$ and $\mathcal{C} = \mathcal{C}(x', e_{uv})$, while case (b) implies $u = s_v(x')$ and $\mathcal{C} = \mathcal{C}(v, e_{xx'})$.

(c) if $x \neq s_{x'}(v)$, $u \neq s_v(x')$ then $\mathcal{C}$ is not isometric.

Proof. The proof can be found in [8]. Note that the cases $e_{uv} \in \mathcal{P}_{xu}$, $e_{uv} \in \mathcal{P}_{xv}$, and $s_x(u) = s_x(v)$ create bridges, thus do not form cycles and need to be excluded. An intuitive explanation of isometric cases is presented in Fig. 3.2. □

In Lemma 3.2 and Fig. 3.2, x' is chosen on the path $\mathcal{P}(xu)$. However, we can also chose x' on the path $\mathcal{P}(xv)$, i.e., letting $x' = s_x(v)$. By doing so, we can find another equivalent representation for an isometric circuit $\mathcal{C}$ by Lemma 3.2.

The connection of different representations can be visualized as a directed graph $\mathcal{G}^\dagger = \mathcal{G}^\dagger(\mathcal{V}^\dagger, \mathcal{E}^\dagger)$, where each vertex in $\mathcal{G}^\dagger$ corresponds to a cycle in the Horton set. If a cycle $\mathcal{C}(x, e_{uv})$ is equivalent to a cycle $\mathcal{C}(y, e_{st})$ by Lemma 3, then an arc is formed from the vertex $\mathcal{C}(x, e_{uv})$ to the vertex $\mathcal{C}(y, e_{st})$ in $\mathcal{G}^\dagger$. It was proved that all representations of an isometric cycle $\mathcal{C}$ exactly correspond to a single connected component in $\mathcal{G}^\dagger$ [8]. The below theorem will greatly improve the efficiency of operations on $\mathcal{G}^\dagger$, in terms of graph storage and searching.

Theorem 1. All representations of an isometric circuit $\mathcal{C}$ in $\mathcal{G}^\dagger$ form a double-linked directed cycle with $|\mathcal{C}|$ vertices.

Proof. It has been proved in [8] that all representations of an isometric cycle $\mathcal{C}$ exactly correspond to a single connected component in $\mathcal{G}^\dagger$. We extend this result in what follows.

There are three possible switches in Lemma 3, i.e., case (1), case (2).(a) and case (2).(b). First, we observe that the switch in each case is mutual: If $\mathcal{C}$ can be switched to $\mathcal{C}'$, then $\mathcal{C}'$ can be switched to $\mathcal{C}$ by the same principle. For case (1), it is straightforward by the fact that $\mathcal{P}_{uv} = \mathcal{P}_{vu}$, so $\mathcal{C}(v, e_{uv}) = \mathcal{C}(u, e_{uv})$. In case (2).(a), we need to prove $\mathcal{C}(x, e_{uv}) = \mathcal{C}(x', e_{uv})$. Starting from $\mathcal{C}(x', e_{uv})$, x is the first vertex on the shortest path from x' to v, i.e. $x = s_{x'}(v)$. It can be verified that $x' = s_x(u)$, so according to Lemma 3.2.(a), $\mathcal{C}(x', e_{uv}) = \mathcal{C}(x, e_{uv})$. In Lemma 3.(2).(b), we prove $\mathcal{C}(x, e_{uv}) = \mathcal{C}(v, e_{xx'})$. Starting from $\mathcal{C}(v, e_{xx'})$, u is the first vertex on the shortest path from v to x', i.e. $u = s_v(x')$. It can be verified that $v \neq s_u(x)$ and $x' = s_x(u)$, so by Lemma 3.(2).(b), $\mathcal{C}(v, e_{xx'}) = \mathcal{C}(x, e_{uv})$. This implies that in $\mathcal{G}^\dagger$, the arcs (i.e., directed edges) are mutual: If there is an arc from $\mathcal{C}$ to $\mathcal{C}'$, then there is an arc from $\mathcal{C}'$ to $\mathcal{C}$ as well.

Second, for an isometric circuit $\mathcal{C}(x, e_{st})$, we can generate exactly two switches because in Lemma 3, the vertex u can be chosen as either s or t, and v as either t or s accordingly. This implies that if a vertex in $\mathcal{G}^\dagger$ is a representation of an isometric circuit, it must have an out-degree $d_{out} = 2$.

Now assume we replace the two arcs mutually connecting $\mathcal{C}$ and $\mathcal{C}'$ in $\mathcal{G}^\dagger$ by an undirected edge. Considering the fact that all representations of an isometric circuit lie in the same connected component of $\mathcal{G}^\dagger$ [8], we conclude: By replacing mutual arc pairs in $\mathcal{G}^\dagger$ as undirected edges, all representations of an isometric circuit constitute a connected subgraph whose vertices have degree of 2, which is a simple cycle. Therefore the directed version is a double-linked directed cycle in $\mathcal{G}^\dagger$.

An isometric circuit has $|\mathcal{C}|$ representations, so the directed cycle has $|\mathcal{C}|$ vertices. □

A visualization of connected components in $\mathcal{G}^\dagger$ is given in Fig. 3.3.

Based on Theorem 1, the storage of $\mathcal{G}^\dagger$ can be compressed to a vector with 2 slots

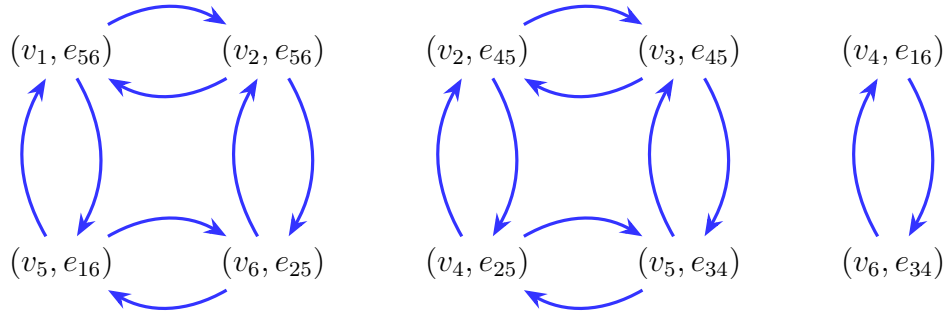


Figure 3.3 A visualization of connected components in $\mathcal{G}^\dagger$. Each isometric circuit $\mathcal{C}$ in $\mathcal{G}$ corresponds to a double-linked directed cycle in $\mathcal{G}^\dagger$ with $|\mathcal{C}|$ vertices. This example is created using the graph in Fig. 2.1, while the cycle representations that do not create links by Lemma 3 are ignored.

reserved for each vertex. We can easily access the adjacent vertices of a given vertex by its index. Besides, a depth-first-search (DFS) [134] on connected components can be simplified as: starting from an arbitrary vertex, keep exploring new vertices until coming back to the start-vertex. This eliminates the use of function recursions or data structures like a stack.

Finally, we characterize the connected components corresponding to isometric cycles by the above simplified DFS. Duplicate representations of an isometric cycle are then removed by keeping only one representation in the connected component. All representations of non-isometric cycles are discarded. The construction of isometric cycles from a Horton set $\boldsymbol{\mathcal{H}}$ (constructed by a consistent APSP) can be achieved with an amortized complexity $O(|\mathcal{V}||\mathcal{E}|)$ [8].

Remark 5. It should be noted that the set of all isometric cycles is a superset to a MCB, but not all MCBs. In other words, even though APSP is chosen to be consistent, a cycle $\mathcal{C}$ in an arbitrary MCB can be non-isometric [100].

3.3.3 Independence Test

To extract a MCB, we sort the set of isometric cycles in non-descending order of weights, and sequentially extract ν linearly independent cycles with the least weights. This procedure is proved to find a MCB [85]. To test the linear independence, we take a circuit as a vector

on $\mathbb{Z}_2^m$ incident on the set of edges, and then evaluate the linear independence algebraically.

Given a (spanning) tree $\mathcal{T}$, let the restricted incidence vector of $\mathcal{C}$ be $\bar{\mathcal{C}} = \mathcal{C}\backslash\mathcal{T}$, i.e. considering the off-tree edges of $\mathcal{T}$ only [50]. It can be shown that the linear independence of a collection of cycles $\{\mathcal{C}_i\}_{i\in N}$ implies the linear independence of the corresponding restricted incidence vectors $\{\bar{\mathcal{C}}_i\}_{i\in N}$, and vice versa (see Theorem 7 in Appendix B for a proof).

Gaussian Elimination Based Approach

Gaussian elimination is a well-exploited technique in graph theory to check the linear independence of incidence vectors [72, 85]. The basic idea is to stack the set of (restricted) incidence vectors as a matrix which will be subsequently reduced to the row echelon form. The incidence vectors are linearly independent if and only if the row echelon form has full row rank. This approach has a cubic complexity in the worst case.

Support Vector Based Approach

Let $\{\mathcal{C}_i\}_{i=1}^{k-1}$ be a set of independent circuits. Given a spanning tree $\mathcal{T}$, let $\{\bar{\mathcal{C}}_i\}_{i=1}^{k-1}$ be the corresponding restricted incidence vectors whose span is a space $\mathbf{C}_{[1:k-1]}$. Denote the orthogonal complementary space as $\mathbf{S}_{[k:\nu]} = \mathbf{C}_{[1:k-1]}^{\perp}$. Let $\{\bar{\mathcal{S}}_i\}_{j=k}^{\nu}$ be a basis of the space $\mathbf{S}_{[k:\nu]}$. The vectors $\{\bar{\mathcal{S}}_i\}_{j=k}^{\nu}$ are called support vectors of $\{\mathcal{C}_i\}_{i=1}^{k-1}$ [99, 100]. Then a circuit $\mathcal{C}_k$ is linearly independent from the set of independent circuits $\{\mathcal{C}_i\}_{i=1}^{k-1}$, if and only if there exists a $\bar{\mathcal{S}}_l$ $(k \le l \le \nu)$, such that $\langle \bar{\mathcal{C}}_k, \bar{\mathcal{S}}_l \rangle = 1$ (see Lemma 10 in Appendix B). If such a $\bar{\mathcal{S}}_l$ is found, then $\{\mathcal{C}_i\}_{i=1}^{k} = \{\mathcal{C}_i\}_{i=1}^{k-1} \cup \mathcal{C}_k$ are a set of independent circuits. The support vectors of $\{\mathcal{C}_i\}_{i=1}^{k}$, i.e. a basis of the space $\mathbf{S}_{[k+1:\nu]} = \mathbf{C}_{[1:k]}^{\perp}$ can be obtained by updating $\{\bar{\mathcal{S}}_i\}_{j=k,j\neq l}^{\nu}$ using $\bar{\mathcal{S}}_l$ [99, 100]. Let

$$\bar{\mathcal{S}}_j' = \begin{cases} \bar{\mathcal{S}}_j & \text{if } \langle \bar{\mathcal{C}}_k, \bar{\mathcal{S}}_l \rangle = 0 \\ \bar{\mathcal{S}}_j + \bar{\mathcal{S}}_l & \text{if } \langle \bar{\mathcal{C}}_k, \bar{\mathcal{S}}_l \rangle = 1 \end{cases},$$

then $\{\bar{\mathcal{S}}_j'\}_{j=k,j\neq l}^{\nu}$ is a set of support vectors for $\{\mathcal{C}_i\}_{i=1}^{k}$.

A direct use of support vectors to check independence can be found in [119]. We invert the process in [119] to accommodate our description. Given a spanning tree $\mathcal{T}$, we can initialize ν independent support vectors by ν off-tree edges, where each support vector contains one off-tree edge [119]. Then at each phase, we evaluate the independence of a new circuit $\mathcal{C}$ by support vectors, which will be subsequently updated if $\mathcal{C}$ is evidenced to be independent by a support vector $\mathcal{S}_l$. Iterate this process until a MCB is found. As in [119], the drawback of this method is that we might need to check many support vectors in order to verify $\langle \mathcal{C}, \mathcal{S}_l \rangle = 1$.

A more sophisticated design is due to Amaldi et al. [10]. The approach is based on the idea that if a circuit $\mathcal{C}$ contains edges that are not used by any selected circuits, then this circuit is independent from the selected ones. If this is the case, we can verify the independence of $\mathcal{C}$ without using any support vector. Moreover, the "new" edges in $\mathcal{C}$ can be used to construct new support vectors. Particularly, there is no need to designate a spanning tree $\mathcal{T}$ and initialize a set of independent support vectors at the beginning of the algorithm. The spanning tree $\mathcal{T}$ is built adaptively by greedily including "new" edges without creating a cycle to maximize the sparsity of $\mathcal{C}$. Let the new edges in $\mathcal{C} \backslash \mathcal{T}$ be $\mathcal{C}_N = \{e_1, \ldots, e_k\}$, then we can identify maximally k independent support vectors, for example, $\mathcal{S}_0 = \{e_1\}$ and $\mathcal{S}_j = \{e_j, e_{j+1}\}$ $(1 \leq j \leq k-1)$. Obviously, $\langle \mathcal{C}, \mathcal{S}_0 \rangle = 1$ and $\langle \mathcal{C}, \mathcal{S}_k \rangle = 0$, which means $\mathcal{S}_0$ is an implicit support vector that evidences $\mathcal{C}$, and there is no need to update $\mathcal{S}_k$ for $\mathcal{C}$. If $\mathcal{C}$ does not contain any new edge, the algorithm checks the existing support vectors by Lemma 10 to verify the independence instead. To speed up the inner product $\langle \mathcal{C}, \mathcal{S} \rangle$, the algorithm maintains $\mathcal{E}_{\mathcal{S}}$ to be the edges used by present support vectors, and $\mathcal{E}^{\circ}$ to be those not anymore because of the update of support vectors ($\mathcal{E}_{\mathcal{S}} \cup \mathcal{E}^{\circ}$ is the set of off-tree edges). At each stage, the edges in both $\mathcal{T}$ and $\mathcal{E}^{\circ}$ can be excluded to increase the sparsity of $\mathcal{C}$.

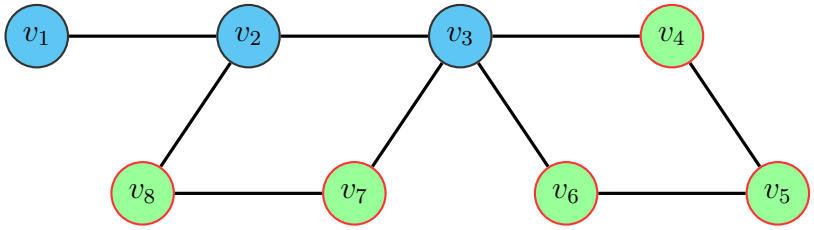
We will use the algorithm by Amaldi et al. [10] to extract a MCB from the set of isometric cycles.

3.3.4 Smoothing Out Vertices of Degree Two

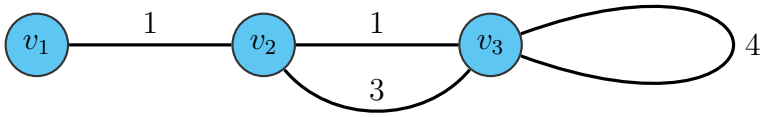
For PGO, the underlying graph is usually sparse. Furthermore, we assume that the sparsity of PGO is positively correlated to the proportion of vertices of degree two in the graph. The sparser the graph is, the more vertices of degree two we have. The vertices of degree two have no contribution to the topology of the graph, thus can be pruned out for the computation of a MCB. The pruning of vertices of degree two, along with the edges incident to them, would greatly reduce the combinatorial complexity of the Horton set.

Algorithmically, we can perform a DFS from a node whose degree is not two. During the search, if a vertex of degree two is detected, we greedily probe along the "degree two chain" until a vertex whose degree is not two is found. Then we replace the "degree two chain" by a new edge (let us name it a "chain edge") whose weight is the accumulated weight along the chain (see Fig. 3.4). The DFS is recursively called at every unvisited vertex whose degree is not two. Finally, after computing a MCB on the reduced graph, the "chain edges" can be replaced back by the corresponding "degree two chains" to obtain a MCB of the original graph.

Remark 6. (Ear Decomposition). The vertices of degree two can also be pruned out using the ear decomposition [133], which requires the graph to be 2-edge connected. A minimum cycle basis algorithm exploiting ear decomposition is provided in [62]. The algorithm in [62] exploiting the idea of feedback vertices to reduce the Horton set, which is encompassed in the concept of isometric cycles [10]. Nevertheless, the set of isometric cycles are much smaller than the set of cycles exploiting the idea of feedback vertices [10]. Actually, an ear decomposition is closely related to a depth-first-search (DFS) of the graph [133]. As a result, the task of pruning vertices of degree two can be achieved by DFS explicitly without computing an ear decomposition as an intermediate step.



(a) Original unweighted graph.



(b) Reduced weighted graph.

Figure 3.4 The original graph (unweighted) and the corresponding reduced graph (weighted) by smoothing out the vertices of degree two. The weight of an edge in the reduced graph is the number of edges it represents in the original graph. Both graphs possess the same cycle structure.

3.3.5 Self Loops and Multiple Edges

Sometimes the graph may contain self-loops or multiple-edges which can be created artificially, or as a consequence of smoothing out vertices of degree two (see Section 3.3.4 and an example in Fig. 3.4). The self-loops and multiple-edges can be easily coped with by describing paths and cycles as a set of edges (instead of vertices) in the MCB algorithm. It is also possible to eliminate all self-loops and multiple-edges for the MCB computation (see Lemma 3.17 and Lemma 3.18 in [100]). However, as shown in the experiments, the computational bottle-neck of the MCB is APSP, while the cost on the independence test is negligible. Thus we retain self-loops and multiple-edges in the graph, and opt to use edges to describe paths and cycles.

3.3.6 LexDijkstra and Parallelism

The bottleneck of the described MCB algorithm is the computation of a consistent APSP (see the experimental results in Table 3.2).

The method described in [84] first compute all-pairs-shortest-distances (APSD) (by using any shortest paths algorithm), then a set of consistent APSP is constructed by choosing the so-called lexicographic shortest path for each pair of vertices (see Definition 2 and Lemma 2), by processing vertex-pairs according to distances (i.e. weight and length) from the shortest to the longest. Obviously, a sorting process is required [84], which can be mitigated by a topological sorting [100]. Besides, although not shown in amortized complexity, the random access to shortest path trees is expensive, in particular for a serial processing (by distances).

The algorithm in [84] is general, applicable for graphs with negative weights and any APSP algorithm. However, for a sparse graph with positive weight, Dijkstra [56] is always the preferable shortest paths algorithm. It can be shown that the lexicographic shortest path for each vertex-pairs can be selected by slightly modifying Dijkstra's update process. The resultant APSP is to run Dijkstra for each vertex, which can be easily parallelized by using a multi-core CPU. We refer to this consistent APSP method as **LexDijkstra**.

Theorem 2. Let $\mathcal{G}$ be an undirected graph with weight $w(e) > 0, \forall e \in \mathcal{E}$. A single-source shortest path tree grown at v can be computed using Dijkstra's algorithm [56]. Let $\mathcal{P}_{uv}$ be a shortest path from u to v. Then for Dijkstra's algorithm, if $w(\mathcal{P}_{uv})$ comes to be the minimum weight available amongst that of all vertices whose shortest path has not been decided yet (such that u is the next vertex to be expanded), all possible paths from u to v with weight $w(\mathcal{P}_{uv})$ have been traversed.

Proof. Let an arbitrary shortest path from u to v with the minimum weight $w(\mathcal{P}'_{uv}) = w(\mathcal{P}_{uv})$ be described by a vertex-edge alternating sequence $\mathcal{P}'_{uv} = \{u \text{ - } e_{us} \text{ - } s, \cdots v\}$. Then there exists a path $\mathcal{P}_{sv} = \mathcal{P}'_{uv} \backslash e_{us}$ from s to v with weight $w(\mathcal{P}_{sv}) < w(\mathcal{P}_{uv})$ because $w(e_{us}) > 0$. When $w(\mathcal{P}_{uv})$ comes as the minimum weight amongst that of all vertices whose shortest path

has not been decided yet, $\mathcal{P}_{sv}$ must have been considered, and a shortest path from s to v must have been found with weight at most $w(\mathcal{P}_{sv})$. Then at the expansion stage of node s, the edge e_{us} has been traversed, so has been the path $\mathcal{P}'_{uv} = \mathcal{P}_{sv} \cup e_{us}$. □

Proposition 2. (LexDijkstra). Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be an undirected graph with weight $w(e) > 0, \forall e \in \mathcal{E}$. A consistent shortest path $\mathcal{P}_{uv}$ can be obtained for each pair of nodes u, v by modifying Dijkstra's update process to choose the lexicographic path in Definition 2.

Proof. By Theorem 2, all shortest paths from u to v with exactly the same minimum weight, which is the case (2) and (3) in Definition 2, will be traversed by Dijkstra's algorithm before the final path $\mathcal{P}_{uv}$ is decided. Thus it would be sufficient to perform a "lexicographic comparison" in Dijkstra's update process according to Definition 2 whenever a new path from u to v is found. By Lemma 2, the APSP comprising lexicographic paths constructed by Definition 2 is consistent. Therefore $\mathcal{P}_{uv}$ constructed by Definition 2 is a consistent path in a consistent APSP. □

In terms of the lexicographic comparison defined in Definition 2, case (1) and case (2) are rather cheap. However, case (3) in the worst case, needs to traverse and compare all the edges in $\mathcal{P}'_{uv}$ and $\mathcal{P}'_{uv}$, which is rather inefficient. To address this issue, we propose the following theorem that greatly reduces the complexity in case (3).

Theorem 3. Let $\mathcal{P}_{uv}$ and $\mathcal{P}'_{uv}$ be two paths from u to v. In LexDijkstra, if the algorithm reaches the case (3) of Definition 2, it just suffices that the algorithm traverses back to the nearest common vertex that shared by $\mathcal{P}_{uv}$ and $\mathcal{P}'_{uv}$.

Proof. Let $\mathcal{P}_{uv}$ and $\mathcal{P}'_{uv}$ be two paths from u to v. Let x be an arbitrary common vertex shared by $\mathcal{P}_{uv}$ and $\mathcal{P}'_{uv}$. Then $\mathcal{P}_{uv}$ and $\mathcal{P}'_{uv}$ can be divided by x into two parts: $\mathcal{P}_{uv} = \mathcal{P}_{ux} + \mathcal{P}_{xv}$, and $\mathcal{P}'_{uv} = \mathcal{P}'_{ux} + \mathcal{P}'_{xv}$. Without loss of generality, let us assume the shortest path tree of LexDijkstra rooted at v. Then by the time comparing paths between u and v, the lexicographic shortest path between x and v must have been found, and is unique,

i.e., $\mathcal{P}'_{xv} = \mathcal{P}_{xv}$. As a result, in case (3) of Definition 2, we have: $\mathcal{P}_{uv} \backslash \mathcal{P}'_{uv} = \mathcal{P}_{ux} \backslash \mathcal{P}'_{ux}$, and $\mathcal{P}'_{uv} \backslash \mathcal{P}_{uv} = \mathcal{P}'_{ux} \backslash \mathcal{P}_{ux}$. Therefore it suffices to compare the paths (i.e., indices of edges) between u and x. Since x is chosen arbitrary, we compare to the nearest shared vertex. □

For the construction of isometric circuits, the algorithm requires random access to the shortest path trees. However, this part can be parallelized without any additional effort.

3.3.7 Complexity

The overall computational time of the proposed MCB algorithm is presented in Table 3.2, running on a quad-core CPU. Table 3.2 shows that the time spending on the independence test is negligible compared with that spending on computing a consistent APSP and constructing the isometric set. Let $\bar{\mathcal{G}}(\bar{\mathcal{V}}, \bar{\mathcal{E}})$ be the reduced graph. Let $m = |\bar{\mathcal{E}}|$, $n = |\bar{\mathcal{V}}|$.

Computing a Consistent APSP

Let us compare the proposed LexDijkstra with the Method in [84]. LexDijkstra: The Dijkstra algorithm has a sorting bottleneck which is usually addressed by a priority queue. In one single Dijkstra run, for each new edge, the operation on the priority queue has a complexity $O(\log n)$. In the worst case, the lexicographic comparison between two paths takes $O(n)$ operations. Therefore, each new edge contributes $O(\log n + n)$ worst case complexity, which results in $O(m(\log n + n))$ operations for one single LexDijkstra run, and $O(nm(\log n + n)$ operations to compute a consistent APSP using LexDijkstra. Method in [84]: By the method in [84], the overall operations used to compute a consistent APSP is $O(nm \log n + n^2 \log n + nm)$, where the term $O(nm \log n)$ accounts for operations to compute an arbitrary APSP based on the Dijkstra algorithm, $O(n^2 \log n)$ for sorting paths according to weights and lengths, and $O(nm)$ for constructing the consistent shortest paths.

By the worst cases complexity, it seems that LexDijkstra does not offer any benefits compared with the method in [84]. However, this is not the cases in practice (see Table 3.2).

The reason is four folds: First, $O(nm)$ in the method [84] cannot be eliminated because it requires random access to all pairs shortest path trees, which is expensive on modern architectures. Second, the method in [84] cannot be run in full parallelism since the sorting term $O(n^2 \log n)$ and processing term $O(nm)$ are sequential operations. Third, $O(n)$ is the worst case complexity for lexicographic comparison, while in practice the operations can be greatly reduced due to the inherent asymmetry in the graph and by Theorem 3 as well. Last but not least, LexDijkstra can run in full parallelism which can take advantage of multi-core CPU architectures.

Constructing the Isometric Set

It takes $O(nm)$ operations to find a single representation for each isometric circuit [8]. However there is a big constant due to the random access to the all pairs shortest path trees, which is implemented as a dense $n \times n$ matrix. While the running time of this part is not major in Table 3.2 compared with the expenses on the consistent APSP, we believe this part can be further improved using a sparse storage for all pairs shortest path trees, given that most of the elements in the dense matrix are redundant because of the consistency of shortest paths.

3.4 Equivalence of VB-PGO and CB-PGO

To begin with, let us restate the least squares based PGO formulation (3.1), i.e., VB-PGO, explicitly by using the relative poses,

$$\{\mathbf{T}_i\}_{i\in\mathcal{V}} = \operatorname{argmin} \sum_{(i,j)\in\mathcal{E}} \|\mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1} \cdot \mathbf{T}_i^{-1}\mathbf{T}_j)\|^2_{\mathbf{\Sigma}_{i,j}} \tag{3.9}$$

which aims to obtain a maximum likelihood (MLE) estimator for the set of poses $\{\mathbf{T}_i\}_{i\in\mathcal{V}}$ using the set of measurements of relative poses $\{\tilde{\mathbf{T}}_{i,j}\}_{(i,j)\in\mathcal{E}}$. Recall that the relative poses is $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j$.

The following result is useful to derive equivalent optimization problems.

Lemma 4. If there exists a bijective mapping $\mathbf{\Phi} : \mathbf{t} \rightarrowtail \mathbf{x}$ such that $\mathbf{x} = \mathbf{\Phi}(\mathbf{t})$, $\mathbf{t} = \mathbf{\Phi}^{-1}(\mathbf{x})$, then the optimization problem with respect to $\mathbf{x}$

$$\min f(\mathbf{x})$$

and the optimization problem with respect to $\mathbf{t}$

$$\min f(\mathbf{\Phi}(\mathbf{t}))$$

are equivalent optimization problems. If $\mathbf{x}^\star$ minimizes $f(\mathbf{x})$, then $\mathbf{t}^\star = \mathbf{\Phi}^{-1}(\mathbf{x}^\star)$ minimizes $f(\mathbf{\Phi}(\mathbf{t}))$, and vice versa.

Proof. Here we prove if $\mathbf{x}^\star$ minimizes $f(\mathbf{x})$, then $\mathbf{t}^\star = \mathbf{\Phi}^{-1}(\mathbf{x}^\star)$ minimizes $f(\mathbf{\Phi}(\mathbf{t}))$. The opposite direction can be proved analogously. If $\mathbf{x}^\star$ minimizes $f(\mathbf{x})$, then $\forall \mathbf{x} \neq \mathbf{x}^\star$, we have $f(\mathbf{x}) > f(\mathbf{x}^\star)$. Assume there exists $\mathbf{t}' \neq \mathbf{t}^\star$ that minimizes $f(\mathbf{\Phi}(\mathbf{t}))$, then we have $f(\mathbf{\Phi}(\mathbf{t}')) < f(\mathbf{\Phi}(\mathbf{t}^\star))$. Since $\mathbf{\Phi}$ is bijective, there exists $\mathbf{x}' = \mathbf{\Phi}(\mathbf{t}')$ such that $f(\mathbf{x}') = f(\mathbf{\Phi}(\mathbf{t}')) < f(\mathbf{\Phi}(\mathbf{t}^\star)) = f(\mathbf{x}^\star)$, which contradicts the fact that $\mathbf{x}^\star$ is the minimizer of $f(\mathbf{x})$. □

3.4.1 Spanning Tree Parameterization and FCB based CB-PGO

The below fact is important for later discussions.

Lemma 5. Let $\mathcal{T}_\circ$ be a spanning tree of $\mathcal{G}$ rooted at the vertex $\circ$. For any vertex k, there exists and only exists one unique path between the vertex $\circ$ and k inside the tree.

Proof. Otherwise, let $\mathcal{P}_{\circ k} \neq \mathcal{P}'_{\circ k}$ be two paths inside $\mathcal{T}_\circ$. Then $\mathcal{P}_{\circ k} \cup \mathcal{P}'_{\circ k}$ constitute a cycle in $\mathcal{T}_\circ$, indicating $\mathcal{T}_\circ$ is no long a tree. Thus $\mathcal{P}_{\circ k} = \mathcal{P}'_{\circ k}$, which proves the uniqueness. □

By Lemma 5, given an anchor vertex $\circ$ and the spanning tree $\mathcal{T}_\circ$ rooted at the vertex $\circ$, we can identify a one-to-one correspondence between a vertex i and the tree path between $\circ$ and i. Denote the tree path on $\mathcal{T}_\circ$ as $\mathcal{P}_{\circ i}$, where $\mathcal{P}_{\circ i} \subseteq \mathcal{T}_\circ$.

Lemma 5 induces a spanning tree parameterization which is a bijective mapping between poses and geometric tree paths. In specific, an arbitrary pose $\mathbf{T}_i$ can be parameterized as $\mathbf{T}_i = \mathbf{T}_\circ \boldsymbol{\mathcal{P}}_{\circ i}$, with $\mathbf{T}_\circ$ being the anchor pose, and $\boldsymbol{\mathcal{P}}_{\circ i}$ being the corresponding geometric tree path from $\circ$ to i. By Lemma 5, the parameterization between $\{\mathbf{T}_i\}_{i\in\mathcal{V}}$ and $\{\boldsymbol{\mathcal{P}}_{\circ i}\}_{i\in\mathcal{V}}$ is bijective, thus satisfying Lemma 4. Using the spanning tree parameterization, the relative poses term $\mathbf{T}_{i,j} = \mathbf{T}_i^{-1}\mathbf{T}_j$ in the VB-PGO formulation writes

$$\mathbf{T}_i^{-1}\mathbf{T}_j = (\mathbf{T}_\circ \boldsymbol{\mathcal{P}}_{\circ i})^{-1}(\mathbf{T}_\circ \boldsymbol{\mathcal{P}}_{\circ j}) = \boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j}.$$

In light of Lemma 4, the equivalent spanning tree parameterization of VB-PGO writes

$$\min \sum_{(i,j)\in\mathcal{E}} \|\mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1} \cdot \boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j})\|^2_{\boldsymbol{\Sigma}_{i,j}}. \tag{3.10}$$

The geometric paths, $\boldsymbol{\mathcal{P}}_{\circ i}$ and $\boldsymbol{\mathcal{P}}_{\circ j}$ are a sequence of relative poses from $\circ$ to i, and $\circ$ to j respectively. By expanding a geometric path as a chain of relative poses, the poses inside the path cancel along the chain, leaving behind only two endpoints of the path, i.e., $\boldsymbol{\mathcal{P}}_{\circ i} = \mathbf{T}_\circ^{-1}\mathbf{T}_i$. This leads to the fact that $\boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j} = (\mathbf{T}_\circ^{-1}\mathbf{T}_i)^{-1}(\mathbf{T}_\circ^{-1}\mathbf{T}_j) = \mathbf{T}_i^{-1}\mathbf{T}_j$, validating the equivalence between (3.9) and (3.10).

Note that a spanning tree in $\mathcal{G}$ induces a fundamental cycle basis (FCB). The next step is to show that the VB-PGO based on the spanning tree parameterization in (3.10) equals to the CB-PGO based on the FCB induced by the exactly same spanning tree. To that end, we apply an explicit variable replacement, by introducing a new variable $\mathbf{M}_{i,j} = \boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j}$, and forcing this relation by constraints:

$$\begin{aligned} &\min \textstyle\sum_{(i,j)\in\mathcal{E}} \|\mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1} \cdot \mathbf{M}_{i,j}\|^2_{\boldsymbol{\Sigma}_{i,j}} \\ &\mathbf{s.t.} \quad \mathbf{M}_{i,j} = \boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j} \quad \forall (i,j) \in \mathcal{E}. \end{aligned} \tag{3.11}$$

Obviously, the constrained optimization problem (3.11) is equivalent to (3.10).

Let the topological paths of $\boldsymbol{\mathcal{P}}_{\circ i}$ and $\boldsymbol{\mathcal{P}}_{\circ j}$ be $\mathcal{P}_{\circ i}$ and $\mathcal{P}_{\circ j}$ respectively. By definition, $\mathcal{P}_{\circ i}$ and $\mathcal{P}_{\circ j}$ are tree-paths on $\mathcal{T}_\circ$. In other words, $\mathcal{P}_{\circ i}$ and $\mathcal{P}_{\circ j}$ only use the edges contained in

$\mathcal{T}_\circ$. For the edge $e_{i,j}$, there are two cases based on whether or not e_{ij} is contained in $\mathcal{T}_\circ$. This two cases are detailed as below using the symmetric difference operation of sets $\oplus$.

1. If the edge e_{ij} is also contained in $\mathcal{T}_\circ$, i.e., $e_{ij} \in \mathcal{T}_\circ$, then $e_{ij} = \mathcal{P}_{\circ i} \oplus \mathcal{P}_{\circ j}$.

2. If the edge e_{ij} is not contained in $\mathcal{T}_\circ$, i.e., $e_{ij} \notin \mathcal{T}_\circ$, then $e_{ij} \oplus \mathcal{P}_{\circ i} \oplus \mathcal{P}_{\circ j}$ is a cycle in $\mathcal{G}$.

In case 1, the relative poses in $\boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j}$ cancels except for the relative poses $\mathbf{T}_{i,j}$. Thus $\mathbf{M}_{i,j} = \boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j}$ admits a trivial identity $\mathbf{M}_{i,j} = \mathbf{T}_{i,j}$. In case 2, the term $\boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j}$ forms a geometric path connecting the pose i and pose j, after canceling the shared relative poses. Let $\boldsymbol{\mathcal{P}}_{ij}$ be the geometric path on the spanning tree connecting i and j. The constraint $\mathbf{M}_{i,j} = \boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j}$ interprets to $\mathbf{M}_{i,j} = \boldsymbol{\mathcal{P}}_{ij}$.

In case 1, it has been shown that for $e_{ij} \in \mathcal{T}_\circ$, we have $\mathbf{M}_{i,j} = \mathbf{T}_{i,j}$. In case 2, where e_{ij} are chords (i.e., off-tree edges), we can safely let $\mathbf{M}_{i,j} = \mathbf{T}_{i,j}$. This is because: (a) physically $\boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j} = \mathbf{T}_i^{-1}\mathbf{T}_j$; (b) we notice that $\mathbf{T}_{i,j}$ $(i,j) = e_{ij} \notin \mathcal{T}_\circ$ will never appear in $\boldsymbol{\mathcal{P}}_{\circ i}$ or $\boldsymbol{\mathcal{P}}_{\circ j}$, thus $\mathbf{T}_{i,j}$ itself in this case is a new variable.

At last, we reach a constrained optimization problem

$$\begin{aligned} &\min \textstyle\sum_{(i,j)\in\mathcal{E}} \|\mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1} \cdot \mathbf{T}_{i,j}\|_{\boldsymbol{\Sigma}_{i,j}}^2 \\ \mathbf{s.t.} \quad &\mathbf{T}_{i,j} = \boldsymbol{\mathcal{P}}_{\circ i}^{-1}\boldsymbol{\mathcal{P}}_{\circ j} = \boldsymbol{\mathcal{P}}_{ij} \quad \forall (i,j) \in \mathcal{E}, (i,j) \notin \mathcal{T}_\circ. \end{aligned} \tag{3.12}$$

The problem (3.12) is a CB-PGO formulation based on the FCB, which is shown to be equivalent to (3.11) by the analysis above. This concludes the equivalence of the VB-PGO in (3.9) and the FCB based CB-PGO in (3.12).

3.4.2 Connected Sum Basis and Network Commutativity

For PGO, all the cycles in the graph must be consistent after optimization. This property is termed "network commutativity" in the field of network science and graph theory [81].

If the group operation describing the relative difference is commutative (i.e., cases in real number synchronizations [141], or rotation synchronizations in 2D [32][34]), the consistency

of the cycles in any cycle basis is sufficient to ensure the consistency of all cycles, i.e., in these cases, the network commutativity can be enforced by any cycle basis. However, for more general scenarios, i.e., for non-planar graphs and non-commutative group operations (for example SE(3)), the cycle basis has to be chosen to be a special basis called "connected sum basis" [81].

Briefly speaking, the "connected sum" requires two simple cycles concatenated to intersect on a single non-trivial path. This is important to derive the consistency of PGO, since the relative poses along the single non-trivial path will cancel by variable substitutions when concatenating two geometric cycles. For example, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two cycles that intersect by a single non-trivial path $\mathcal{P}_*$. The corresponding geometric cycles can be written as: $\boldsymbol{\mathcal{C}}_1 : \ \boldsymbol{\mathcal{P}}_{1a}\boldsymbol{\mathcal{P}}_*\boldsymbol{\mathcal{P}}_{1b} = \mathbf{I}$, and $\boldsymbol{\mathcal{C}}_2 : \ \boldsymbol{\mathcal{P}}_{2a}\boldsymbol{\mathcal{P}}_*\boldsymbol{\mathcal{P}}_{2b} = \mathbf{I}$. Then the modulo 2 summation on $\mathbb{GF}$, $\mathcal{C}_1 + \mathcal{C}_2$, or equivalently the symmetric difference $\mathcal{C}_1 \oplus \mathcal{C}_2$, which will eliminate the edges in $\mathcal{P}_*$, corresponds exactly to a variable elimination by canceling $\boldsymbol{\mathcal{P}}_*$ in $\boldsymbol{\mathcal{C}}_1$ and $\boldsymbol{\mathcal{C}}_2$. To show this, we rewrite $\boldsymbol{\mathcal{C}}_1$ as $\boldsymbol{\mathcal{P}}_* = \boldsymbol{\mathcal{P}}_{1a}^{-1}\boldsymbol{\mathcal{P}}_{1b}^{-1}$ and substitute $\boldsymbol{\mathcal{P}}_*$ into $\boldsymbol{\mathcal{C}}_2$ to eliminate $\boldsymbol{\mathcal{P}}_*$, which yields a new geometric cycle $\boldsymbol{\mathcal{C}}_{12} : \ \boldsymbol{\mathcal{P}}_{2a}\boldsymbol{\mathcal{P}}_{1a}^{-1}\boldsymbol{\mathcal{P}}_{1b}^{-1}\boldsymbol{\mathcal{P}}_{2b} = \mathbf{I}$. Note the geometric cycle $\boldsymbol{\mathcal{C}}_{12}$ exactly corresponds to the topological cycle $\mathcal{C}_1 + \mathcal{C}_2$ (or equivalently $\mathcal{C}_1 \oplus \mathcal{C}_2$). If all the cycles in the graph can be generated by connected sums from cycles in a basis, then all geometric cycles can be generated from the geometric cycles in the basis by variable substitutions. This explains why a connected sum basis is critical to ensure the consistency of PGO.

When speaking cycles, we distinguish between two concepts: (1) simple cycles (or circuits), i.e., 2 regular connected subgraphs; (2) general cycles, i.e., even-degree connected subgraphs. A general cycle, i.e., an even-degree connected subgraph, can be expressed as an edge-disjoint union of simple cycles. Thus if all the simple cycles which construct the even-degree connected subgraph are consistent, then the even-degree connected subgraph is consistent automatically. Therefore, to ensure the consistency of all cycles of a graph, it suffices to ensure the consistency of all simple cycles in the graph. Note that cycles in a cycle basis are always simple.

In this section, the cycles and paths are considered as subgraphs, which are characterized by a set of vertices and edges. The symmetric difference $\oplus$ of sets and the summation $+$ on $\mathbb{GF}$ are defined on the induced edge set as usual. Note that even though the edge set is sufficient to characterize a cycle/path, the vertex set is important to confine the concepts "compatible cycles" and "connected sum" to the cases of simple cycles. The following definitions are in line with the ones used in [81].

Definition 3 (Compatible cycles)**.** Two simple cycles $\mathcal{C}_1$ and $\mathcal{C}_2$ are said to be compatible, if $\mathcal{C}_1$ and $\mathcal{C}_2$ intersect by a single non-trivial path, i.e., a path with at least one edge. The vertices shared by $\mathcal{C}_1$ and $\mathcal{C}_2$ must be contained in the path intersected.

The summation (or symmetric difference) of two compatible cycles is again a simple cycle. This is different from the summation in general cases, where the resulting cycle can be general, i.e., even-degree subgraphs. To emphasis, the summation of two compatible cycles is dubbed as connected sum.

Definition 4 (Connected sum)**.** Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two simple cycles. If $\mathcal{C}_1$ and $\mathcal{C}_2$ are compatible, the summation on $\mathbb{GF}$ (i.e., modulo 2 sum) $\mathcal{C}_1 + \mathcal{C}_2$, is called a connected sum.

If two simple cycles $\mathcal{C}_1$ and $\mathcal{C}_2$ are compatible, their summation (i.e., connected sum) is a simple cycle $\mathcal{C}_{12} = \mathcal{C}_1 + \mathcal{C}_2$. Now if we have another simple cycle, saying $\mathcal{C}_3$, and if $\mathcal{C}_3$ is compatible with $\mathcal{C}_{12}$, then these two cycles can be summed up by a connected sum again, to generate a new simple cycle $\mathcal{C}_{123} = (\mathcal{C}_1 + \mathcal{C}_2) + \mathcal{C}_3$. This brings us to the concept of connected sum sequence.

Definition 5 (Connected sum sequence)**.** A sequence of simple cycles $\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_m$ is said to be a connected sum sequence if the partial sum $\mathcal{C}_{1,2,\ldots k} = ((\mathcal{C}_1 + \mathcal{C}_2) + ...) + \mathcal{C}_k$ is compatible with the next cycle $\mathcal{C}_{k+1}$, for all $k = 1, \cdots m-1$.

Definition 6 (Connected sum closure)**.** Given a set of simple cycles, saying $\boldsymbol{\mathcal{S}}$, the connected sum closure of $\boldsymbol{\mathcal{S}}$ is the set of all simple cycles that are generated by the connected sums of sequences from $\boldsymbol{\mathcal{S}}$. Denote the connected sum closure of $\boldsymbol{\mathcal{S}}$ to be $\rho(\boldsymbol{\mathcal{S}})$.

Since the connected sum closure $\rho(\boldsymbol{\mathcal{S}})$ of a set of simple cycles $\boldsymbol{\mathcal{S}}$ is again a set of simple cycles, we can repeat this operation, and recursively define $\rho^0(\boldsymbol{\mathcal{S}}) = \boldsymbol{\mathcal{S}}$, $\rho^1(\boldsymbol{\mathcal{S}}) = \rho(\boldsymbol{\mathcal{S}})$, $\rho^2(\boldsymbol{\mathcal{S}}) = \rho(\rho(\boldsymbol{\mathcal{S}}))$, and $\rho^k(\boldsymbol{\mathcal{S}}) = \rho(\rho^{k-1}(\boldsymbol{\mathcal{S}}))$.

Let $\boldsymbol{\mathcal{S}}$ be a set of simple cycles of a graph $\mathcal{G}$. Let us further denote the set of all simple cycles of $\mathcal{G}$ as $\mathbf{Cyc}(\mathcal{G})$. Then we have

$$\boldsymbol{\mathcal{S}} = \rho^0(\boldsymbol{\mathcal{S}}) \subseteq \rho^1(\boldsymbol{\mathcal{S}}) \subseteq \rho^2(\boldsymbol{\mathcal{S}}) \subseteq \cdots \subseteq \mathbf{Cyc}(\mathcal{G}).$$

Definition 7 (Connected sum generating set)**.** For a set of simple cycles $\boldsymbol{\mathcal{S}}$ on $\mathcal{G}$, if $\rho^k(\boldsymbol{\mathcal{S}}) = \mathbf{Cyc}(\mathcal{G})$ for some k, we say $\boldsymbol{\mathcal{S}}$ is a connected sum generating set.

Definition 8 (Connected sum basis)**.** A cycle basis $\mathcal{B}$ of a graph $\mathcal{G}$ is a connected sum basis if it is a connected sum generating set, that is $\rho^k(\mathcal{B}) = \mathbf{Cyc}(\mathcal{G})$ for some k.

When studying the properties of cycle bases of graphs, it often suffices to confine to the concept of 2-connected graphs. A graph $\mathcal{G}^*(\mathcal{V}^*, \mathcal{E}^*)$ is said to be 2-connected if for any vertex $x \in \mathcal{V}^*$, $\mathcal{G}^\star \backslash x$ (by removing x and edges that use x as endpoints) is connected. In other words, a graph $\mathcal{G}^*$ is 2-connected if there exists (at least) two vertex-disjoint paths connecting any two vertices in $\mathcal{G}^*$. A graph is 2-connected if and only if it has an (open) ear decomposition [156]. An (open) ear decomposition of $\mathcal{G}^*$ is a partition of $\mathcal{G}^*$ into a sequence of pairwise edge disjoint subgraphs $\mathcal{C}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n$, where $\mathcal{C}_0$ is a simple cycle, and each $\mathcal{P}_i$ is a non-trivial path (called an ear). Let $\mathcal{G}_0^* = \mathcal{C}_0$, and $\mathcal{G}_i^* = \mathcal{C}_0 \cup \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_i$ for $i = 1, 2, \ldots n$. Then the sequence $\mathcal{C}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n$ is called an ear decomposition of $\mathcal{G}^*$ if $\mathcal{G}^* = \mathcal{G}_n^*$ and $\mathcal{P}_i^*$ intersects $\mathcal{G}_{i-1}^*$ only with the endpoints of $\mathcal{P}_i^*$. The ear decomposition admits a cycle basis called "ear basis", $\mathcal{B} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots \mathcal{C}_n\}$, where $\mathcal{C}_i$ $(i = 1, 2, \ldots n)$ is constructed by $\mathcal{P}_i$ and a non-trivial path in $\mathcal{G}_{i-1}^*$ connecting the endpoints of $\mathcal{P}_i$.

Note that for a graph that is not 2-connected, simple cycles cannot span different 2-connected blocks of the graph. Moreover, the union of cycle bases of each 2-connected block is a cycle basis for the entire graph.

Lemma 6 ([81])**.** For any 2-connected undirected graph, the ear basis constructed by an ear decomposition is a connected sum basis.

For an general undirected graph, the union of the ear basis of its each 2-connected block forms a connected sum basis of the graph. Thus the following conclusion is also true [81].

Lemma 7 ([81])**.** Every undirected graph has a connected sum basis.

Lemma 7 shows the universal existence of connected sum bases for general undirected graphs. However, until now, the community does not provide characterizations of connected sum bases, in terms of the most commonly used cycle bases like fundamental cycle basis (FCB), or minimum cycle basis (MCB).

In Section 3.4.1, we have shown that the FCB based CB-PGO is equivalent to the VB-PGO. Now we consolidate this result by showing that a fundamental cycle basis (i.e., FCB) is a connected sum basis, by showing that it is an ear basis. Therefore the consistency of cycles in a FCB is sufficient to ensure the consistency of all the cycles in the graph. To prove this result, it suffices to show the property on a 2-connected undirected graph, while for a general graph, we can partition it into several 2-connected blocks.

Let us consider a 2-connected graph $\mathcal{G}^*$. Let $\mathcal{T}^*$ be a spanning tree of $\mathcal{G}^*$. Denote the fundamental cycle basis of $\mathcal{G}^*$ constructed by $\mathcal{T}^*$ to be

$$\mathcal{B}_{FCB} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{\nu^*}\}$$

with ν^* being the dimension of its cycle space (i.e., its cyclomatic number). Then let us define a sequence of subgraphs $\mathcal{G}_i^*$ $(i = 1, \ldots, \nu^* - 1)$ using the cycles in $\mathcal{B}_{FCB}$,

$$\mathcal{G}_i^* = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_i.$$

Given the 2-connected graph $\mathcal{G}^*$, arrange the cycles in $\mathcal{B}_{FCB}$ such that $\mathcal{C}_{i+1}$ $(i = 1, \ldots, \nu^* - 1)$ intersects with $\mathcal{G}_i^*$, i.e., $\mathcal{G}_i^* \cap \mathcal{C}_{i+1} \neq \emptyset$. Then each $\mathcal{G}_i^*$ $(i = 1, \ldots, \nu^* - 1)$ is a connected subgraph by construction. This construction is possible by Theorem 3 in [58].

Below lemma validates the uniqueness of the spanning tree path.

Lemma 8. Given a spanning tree of a connected graph, there exists and only exists one unique spanning tree path between any two distinct vertices in the graph.

Proof. The proof is analogous to that of Lemma 5. □

Theorem 4. Given the graph $\mathcal{G}_i^*$ and the spanning tree $\mathcal{T}^*$, for any two vertices x and y in $\mathcal{G}_i^*$, the unique spanning tree path between x and y is contained in $\mathcal{G}_i^*$.

Proof. This can be proved by induction. For $\mathcal{G}_1^* = \mathcal{C}_1$, the conclusion is true. Let us assume the result is true for $\mathcal{G}_{i-1}^*$. Let x be a vertex in $\mathcal{G}_{i-1}^*$ and y be a vertex in $\mathcal{C}_i$. Let v be a vertex in $\mathcal{G}_{i-1}^* \cap \mathcal{C}_i$. Then for x and v in $\mathcal{G}_{i-1}^*$, there is a spanning tree path $\mathcal{P}_{xv}$. $\mathcal{C}_i$ is a fundamental cycle, then for y and v in $\mathcal{C}_i$, there is a spanning tree path $\mathcal{P}_{vy}$. Therefore, for the subgraph $\mathcal{G}_i^* = \mathcal{G}_{i-1}^* \cup \mathcal{C}_i$, there is a spanning tree path $\mathcal{P}_{xy} = \mathcal{P}_{xv} \oplus \mathcal{P}_{vy}$. This completes the induction and concludes the proof. □

Now we are set to state the following result.

Theorem 5. The fundamental cycle basis $\mathcal{B}_{FCB}$ to the 2-connected graph $\mathcal{G}_i^*$ is an ear basis.

Proof. To prove, it suffices to show that $\mathcal{C}_{i+1}$ can be decomposed as an ear of $\mathcal{G}_i^*$ plus a path contained in $\mathcal{G}_i^*$.

Let e_{uv} be the off-tree edge on $\mathcal{C}_{i+1}$. Then $e_{uv} \in \mathcal{C}_{i+1}$ and $e_{uv} \notin \mathcal{G}_i^*$. Starting from the two endpoints of e_{uv} and traversing along $\mathcal{C}_{i+1}$, we can find two vertices x and y that first time intersect with $\mathcal{G}_i^*$. By doing so, we obtain a path $\mathcal{P}_{xy}$ such that $\mathcal{P}_{xy} \subset \mathcal{C}_{i+1}$ and $\mathcal{P}_{xy} \cap \mathcal{G}_i^* = \{x, y\}$. Then $\mathcal{P}_{xy}$ is an ear to $\mathcal{G}_i^*$.

Recall that cycles in a cycle basis are simple, so $\mathcal{C}_{i+1}$ can be decomposed as $\mathcal{C}_{i+1} = \mathcal{P}_{xy} \cup \mathcal{P}'_{xy}$, with $\mathcal{P}_{xy} \cap \mathcal{P}'_{xy} = \{x, y\}$. Given the fact that $\mathcal{C}_{i+1}$ is a fundamental cycle, and the off-tree edge e_{uv} is contained in $\mathcal{P}_{xy}$, we conclude that $\mathcal{P}'_{xy}$ consists of spanning tree edges only, i.e., $\mathcal{P}'_{xy}$ is the spanning tree path between x and y, which is unique by Lemma 8. Moreover, because x and y lies in $\mathcal{G}_i^*$, by Theorem 4, the unique spanning tree path between x and y, namely $\mathcal{P}'_{xy}$, is contained in $\mathcal{G}_i^*$.

Therefore $\mathcal{C}_{i+1} = \mathcal{P}_{xy} \cup \mathcal{P}'_{xy}$, where $\mathcal{P}_{xy}$ is an ear to $\mathcal{G}_i^*$ and $\mathcal{P}'_{xy}$ is contained in $\mathcal{G}_i^*$.

By induction, $\mathcal{B}_{FCB}$ is an ear basis. □

By Theorem 5, we conclude that for a 2-connected graph, a fundamental cycle basis is an ear basis, which means it is also a connected sum basis by Lemma 6. For a general undirected graph, since the collection of cycle bases in its each 2-connected block is a cycle basis of the undirected graph, we conclude that:

Theorem 6. A fundamental cycle basis of an undirected graph is a connected sum basis.

Theorem 6 means that forcing the consistency of cycles in a FCB is sufficient to ensure the consistency of all the cycles in a graph. This result is in accordance with the analysis in the previous section, i.e., Chapter 3.4.1.

Now we are left with the question: Can we prove a MCB to be a connected sum basis by using the idea of ear decompositions on each 2-connected block of the graph? The author perceive this is not true. This is because an ear basis is a weakly fundamental cycle basis [109]. While a MCB can be computed in polynomial time (i.e. cubic time), if this conjecture is true, then we have just got a minimum weakly fundamental cycle basis algorithm in polynomial time. However the minimum weakly fundamental cycle basis problem is consider as in APX hardness [100].

It seems that we cannot prove a MCB to be a connected sum basis via the construction of ear bases. This leaves us to the next question: **Are there any other type of connected sum bases, rather than ear bases?** And the final question is: **Is a MCB a connected sum basis?**

Unfortunately, by the time writing this thesis, the author won't be able to provide an assertive answer to these questions, thus leaving these questions open to be further investigated in the research community. Hope there will be an answer in the near future.

Nonetheless, the author hold a strong belief that a MCB is a connected sum basis. Because, the numerical experiments in Chapter 3.7 show that MCB based CB-PGO can

attain the same vertex-based objective value as VB-PGO. If some cycles are inconsistent, the vertex estimates derived from the spanning tree are not optimal, thus the objective values reflected in the VB-PGO formulation will be larger. However this is not the case by our observations.

The second support is that, the set of isometric cycles (i.e., geodesic cycles) can build all the cycles of a graph by the connected sum operation [98]. It has been shown that the set of isometric cycles is a tight superset of a MCB [8, 10]. Therefore in the worst case, if a MCB is not a connected sum basis, the MCB still provide a strong guarantee on the global consistency of all cycles, due to its closeness to the set of isometric cycles.

Last but not least, for synchronization problems with commutative group operations, any cycle basis is sufficient to ensure the consistency of all cycles. In these cases, the proposed MCB algorithm can be used without concerns.

3.5 Discussions

3.5.1 Observability

We propose to define the observability in the maximum likelihood estimation (MLE) as:

Definition 9. (Observability of MLE [93]) Let $\mathcal{M}^d$ be a manifold of dimension d. A noise-free system $\mathbf{Z} = \mathfrak{F}(\mathbf{X}) : \mathcal{M}_1^{d_1} \rightarrowtail \mathcal{M}_2^{d_2}$, is locally observable at $\mathbf{X}_0 \in \mathcal{M}_1^{d_1}$ if there is a neighborhood of $\mathbf{X}_0$, denoted by $\mathbb{U}_{\mathbf{X}_0}$, such that

$$\forall \mathbf{X} \in \mathbb{U}_{\mathbf{X}_0},\ \mathbf{X} \neq \mathbf{X}_0,\ \text{we have } \mathfrak{F}(\mathbf{X}) \neq \mathfrak{F}(\mathbf{X}_0).$$

By Definition 9, it is easy to verify that VB-PGO in (3.1) is unobservable. Because given a solution $\mathbf{X}_0 \triangleq \{\mathbf{T}_i\}_{i\in\mathcal{V}}$ and any neighborhood $\mathbb{U}_{\mathbf{X}_0}$ around $\mathbf{X}_0$, by the fact that a Lie group is a continuous group, we can always find a shifted solution $\mathbf{X} \triangleq \{\mathbf{T}'\mathbf{T}_i\}_{i\in\mathcal{V}}$, $\mathbf{X} \in \mathbb{U}_{\mathbf{X}_0}$, which yields exactly the same measurements (see Remark 2).

This result coincides with the Fisher information matrix (FIM) based observability tool (see Remark 7).

Now we extend Definition 9 to estimation problems with constraints, i.e., a noise-free sytem $\mathbf{Z} = \mathfrak{F}(\mathbf{X}) : \mathcal{M}_1^{d_1} \rightarrowtail \mathcal{M}_2^{d_2}$ with constraints $\mathfrak{G}(\mathbf{X}) = \mathbf{I} : \mathcal{M}_1^{d_1} \rightarrowtail \mathcal{M}_3^{d_3}$. If the constraint forms a submanifold $\mathcal{M}_4^{d_4}$ embedded on $\mathcal{M}_1^{d_1}$, then we can reduce the constrained MLE to an unconstrained MLE, and verify the observability by Definition 9. The basic tool is the so-called submersion theorem (Proposition 3.3.3 in [4]): If $\mathfrak{G}$ is smooth, and $\mathbf{I}$ is a regular value of $\mathfrak{G}$ (i.e. the rank of $\mathfrak{G}$ is d_3 for each point in $\mathbf{Y} \in \mathcal{M}_1^{d_1}$ satisfying $\mathfrak{G}(\mathbf{Y}) = \mathbf{I}$), then

$$\mathcal{M}_4 = \{\mathbf{X} \mid \mathbf{X} \in \mathcal{M}_1^{d_1},\ \mathfrak{G}(\mathbf{X}) = \mathbf{I},\ \mathbf{rank}(\mathfrak{G}) = d_3\}$$

admits a differential structure, and $\mathcal{M}_4^{d_4}$ is an embedded submanifold of $\mathcal{M}_1^{d_1}$, with dimension $d_4 = d_1 - d_3$. If a set of constraints is "redundant", we can verify a submanifold by the subimmersion theorem (Proposition 3.3.4 in [4]).

Finally let us examine the case of CB-PGO. Let $\mathcal{M}_1^{d_1}$ be $\mathrm{SE}(3)^m$. The ν constraints clearly satisfy the submersion theorem, thus the constraints admit a submanifold $\mathcal{M}_4^{d_4}$ embedded in $\mathcal{M}_1^{d_1}$. The noise-free system, i.e., the measurement function of relative poses $\mathbf{Z} = \mathfrak{F}(\mathbf{X}) = \mathbf{X}$, with $\mathbf{X} \triangleq \{\mathbf{T_k}\}_{\mathbf{k}\in\mathcal{E}}$, is bijective on $\mathcal{M}_1^{d_1}$ thus its restriction to $\mathcal{M}_4^{d_4}$ is also bijective. Therefore, taking CB-PGO as a MLE on $\mathcal{M}_4^{d_4}$, we verify CB-PGO to be observable by Definition 9.

Remark 7. In estimation theory, a common practice is to define the observability of an estimation problem as the invertibility of the FIM [18], (also see [12, 155] for some applications in robotics). While FIM being a statical tool, which is related to a specific noise model (Gaussian etc.), the deterministic definition of observability for MLE in Definition 9 has been shown to be equivalent to FIM based observability as long as the probability density function has a continuous derivative [93].

Remark 8. The covariance matrix (whose inversion is FIM) of CB-PGO can be computed in closed form (see [15, 73, 118]). However, this covariance matrix is always rank deficient

because of the over-parameterization in (3.3), namely $|\mathcal{E}| > \nu$. This does not mean CB-PGO is unobservable. To apply FIM based tool correctly, we have to obtain FIM for $\mathcal{M}_4^{d_4}$ in its Euclidean space via an atlas [4], instead of using FIM on $\mathcal{M}_1^{d_1}$. Nevertheless, we can verify the observability by Definition 9, without explicitly assigning the atlas.

3.5.2 Jacobian Matrix Design: MCB and Invariance

In CB-PGO, the entry in the Jacobian matrix (Eq. (A.2) in Appendix A.2) takes the form of $\sigma(\mathbf{k}^c)\mathbf{Ad}\left(\boldsymbol{\mathcal{P}}(\alpha(\sigma(\mathbf{k}^c)))\right)$, with $\boldsymbol{\mathcal{P}}(\cdot)$ being a geometric path inside the cycle. Let the corresponding topological path be $\mathcal{P}(\cdot)$. Ideally, we want $\mathcal{P}(\cdot)$ to be as short as possible, so that less errors will be accumulated in $\boldsymbol{\mathcal{P}}(\cdot)$, and the Jacobian can more accurately capture the local structure at the linearization point. By using a MCB, the average length of $\mathcal{P}(\cdot)$ is minimized along with the overall length of cycles. Therefore CB-PGO based on a MCB can be expected to perform better than that based on cycle bases like a FCB. This is true as will be experimentally validated in Fig. 3.6 and Fig. 3.5.

In CB-PGO, we can minimize the linearization errors inside the Jacobian matrix by using a MCB instead of a FCB. The idea of having a Jacobian matrix less relevant to linearization errors has been exploited in the context of Lie group estimation with "invariance" [21, 22, 23, 44, 157] as well. In brief, invariance works by choosing a special Lie group parameterization (i.e., group affine [23]) that the Jacobian matrix obtained via linearization at a certain point is merely related to some "error-states" rather than the linearization point directly. As a result the left/right-hand Jacobian in the BCH formula with respect to the error-states is eliminated. This technique can significantly improve the convergence of the estimation problem, especially when facing large noise scenarios.

3.5.3 Convergence Rate

Regarding constrained optimization, one of the major concerns is its convergence rate. However, given that CB-PGO in (3.3) is essentially an equality constrained least squares optimization problem, techniques like SQP can attain quadratic/superlinear convergence [26]. Here we briefly review some work in this line to confirm the claim. Experimental validations are provided in Fig. 3.5.

For equality constrained least squares optimization, it has been shown in [149] that a quadratic convergence rate can be obtained by applying Newton's method (an iterative method to solve a nonlinear equation [129]) to calculate a critical point of the corresponding Lagrangian function. This extends the quadratic convergence of Newton's method used in unconstrained optimization to constrained cases by SQP. Methods based on approximate Lagrangian Hessians can attain superlinear convergence [27, 46, 66].

In the context of least squares, the Gauss-Newton method yields quadratic convergence if the residual error at the minimum is zero [115]. The same conclusion stands in the case of equality constrained least squares, which was proved in [145]. This convergence result implies that SQP by linearizing the cost function and constraints (like in (3.4)) can have basically similar convergence as the Gauss-Newton method.

These justifications are mostly true if the algorithm works around a minimum. In practice, if the noise level in relative poses is reasonable, CB-PGO initialized by the measurements of relative poses is close to the ground-truth. Therefore CB-PGO can have a better convergence (compared with VB-PGO initialized by odometry) because the working point is closer to a minimum and the Hessian matrix is more accurate.

3.6 Implementation Details

The most well-known open-source implementation of a MCB comes from Dimitrios Michail [119], based on the LEDA graph library. However, the code is not maintained any more

for recent LEDA versions, or Ubuntu systems. The other competitive implementations, for example [10, 62], are not available in open-source. In contrast, our implementation is freely available to the research community, and can be easily used in other MCB related problems.

We implement the MCB algorithm described in Section 3.3 from scratch with C++ Standard Library and C++ Standard Template Library (STL) only. For Dijkstra, we use the priority-queue implementation of STL. We use a dense square matrix with backward pointers to parent vertices to describe the APSP trees. For set operations on support vectors and cycles, sorted sparse vectors are used instead of binary search trees, which is faster by our experiments. OpenMP is used to parallelize CPU computations on multiple cores.

Both CB-PGO and VB-PGO are implemented on an open-source graph optimization library, i.e. SLAM++ [92], which provides the basic operations on block matrices and Cholesky factorization. In SLAM++, we use the default block Cholesky based linear solver, and the approximate minimum degree ordering algorithm (AMD) [11] to reduce the fill-in. Both CB-PGO and VB-PGO use the same Lie group implementation, to avoid the impact of latent numerical round-off errors.

For a fair comparison, we implement chordal initialization [41, 117] as an alternative initialization technique, using Block Cholesky factorization. Instead of initializing rotational part of poses only [41], we reestimate the translational part as well. This will give an accurate initial objective value for the chordal based methods. An initialization of relative poses is obtained by recalculating relative poses with pose estimates, which can be used to initialize CB-PGO.

Noting that for a batch algorithm, we only need to run MCB and AMD once, which will be followed by several iterations of linearization, Cholesky factorization, and state updates.

3.7 Experimental Results

In this paper, the experiments are carried out by a laptop equipped with a quad-core CPU, Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz × 4, running on Ubuntu 16.04 LTS.

We will use four real datasets and four simulated datasets as standard benchmarks. INTEL_P, Intel and MITb are obtained by processing the raw measurements from wheel odometry and laser range finder [34]. Raw data are available at [1]. KITTI is a pose graph generated by the proSLAM framework [144] from the vision benchmark dataset [70]. The four simulated datasets and their creators are: Manhattan [127], City10k [95], Sphere2500 [95], and Torus10000 [95].

We will use the cycle ratio of a graph, defined as $\nu/|\mathcal{E}|$, to benchmark the graph sparsity. In graph simulations, we regard the local minimum computed by using the ground-truth initialization as the global minimum. Let the objective value at the global minimum be $f^\star$. For any computed solution, if its objective value f satisfies $|f/f^\star - 1| < 0.01$, the solution is counted as a success. The number of successes divided by the number of Monte Carlo runs is defined as the success rate. For CB-PGO, at each iteration, we firstly calculate a pose configuration through odometry using the current estimate of relative-poses. Then we substitute the pose configuration to the cost function of VB-PGO, and use the obtained cost as the objective value of CB-PGO.

In this section, the VB-PGO initialized by odometry is denoted as VB, while CB-PGO initialized by the measurements of relative poses is denoted as CB. We distinguish CB-PGO techniques based on the MCB and FCB by denoting them as CB-MCB, CB-FCB respectively. Besides the "natural" initialization for VB-PGO and CB-PGO, we use chordal initialization to bootstrap the rotational part [41]. To evaluate the objective value after chordal initialization, we recompute the optimal translational estimate after the rotational initialization. The chordal bootstrapped VB, and CB variants are termed as VB-Chord, CB-MCB-Chord and CB-FCB-Chord respectively.

The maximal iterations are set to 50, and we stop the algorithm if the perturbation (i.e., state updates) has a norm less than 0.001. For CB-PGO, we also ensure the constraint residual norm to be less than 0.001.

3.7.1 MCB

The timing statistics of the proposed MCB is reported in Table 3.2. The proposed MCB algorithm consists of three major parts: consistent-APSP, isometric set construction (Section 3.3.2), and independence test (Section 3.3.3). The construction of Horton set (Section 3.3.1) requires a APSP, while its reduction to the isometric set requires a consistent-APSP. In Table 3.2, Dijkstra is used to compute a APSP. The consistent-APSP can be computed via the method in [84], or by LexDijkstra (proposed). All the timings regarding these two consistent-APSP algorithms are presented. To benchmark the overall effectiveness of the proposed MCB, we compare with the state-of-the-art open source implementation in [119]. In Table 3.2, the symbol "-" in the "parallel" row means that the corresponding computation cannot be parallelized.

In Table 3.2, the proposed MCB algorithm is at least as fast as the state-of-the-art in [119]. Actually the proposed MCB is much faster for sparse graphs (due to the pruning of degree-two vertices), and slightly faster for most dense graphs, except for the Sphere dataset. This is due to the high-symmetry of the graph which results to many lexicographical comparisons. The implementation in [119] fails at the KITTI dataset, because it cannot handle parallel edges. In addition, Table 3.2 clearly shows the advantage of using the proposed LexDijkstra to compute a consistent-APSP, since it avoids the sequel bottleneck in [84]. In the worst case, LexDijkstra accounts for 1.5 times the timing of the pure Dijkstra (APSP), which we believe is already close to the lower borderline. Note that this timing will double without Theorem 3.

In Table 3.2, the computation is parallelized by a quad-core CPU; however these timings

Table 3.2 The Computational Time of LexDijkstra and MCB on a Quad-core CPU. Time in Seconds.

		Consistent-APSP					Isometric	Indep.	Proposed	Michail [119]
		Method in [84]				LexDijkstra				
		Dijkstra	Sorting	Processing	Overall					
MITb	Seq.	1.17e-4	6.23e-5	5.97e-5	2.39e-4	1.25e-4	5.76e-5	7.24e-6		
	Par.	6.50e-4	-	-	7.72e-4	7.20e-4	4.61e-5	-	8.45e-4	2.69e-3
INTEL_P	Seq.	2.42e-3	5.06e-4	8.66e-4	3.79e-3	2.79e-3	9.01e-4	5.35e-5		
	Par.	1.78e-3	-	-	3.15e-3	1.31e-3	5.46e-4	-	2.11e-3	2.46e-2
KITTI	Seq.	1.36e-3	6.50e-4	1.41e-3	3.42e-3	2.32e-3	1.13e-3	9.49e-5		
	Par.	2.63e-3	-	-	4.69e-3	2.74e-3	7.34e-4	-	3.94e-3	fail
Intel	Seq.	3.52e-2	1.26e-2	2.66e-2	7.44e-2	5.83e-2	1.49e-2	1.88e-4		
	Par.	1.43e-2	-	-	5.35e-2	2.45e-2	8.47e-3	-	3.36e-2	7.95e-2
Manhattan	Seq.	0.495	0.250	0.511	1.26	0.623	0.211	5.66e-4		
	Par.	0.187	-	-	0.948	0.241	0.105	-	0.348	0.598
Sphere2500	Seq.	0.469	0.202	0.601	1.27	2.03	0.215	2.45e-4		
	Par.	0.187	-	-	0.990	0.766	0.116	-	0.883	0.309
City10k	Seq.	7.74	3.45	11.1	22.3	11.7	4.69	6.25e-3		
	Par.	2.88	-	-	17.4	4.53	2.71	-	7.26	8.42
Torus10k	Seq.	8.91	3.64	13.6	26.2	14.3	5.93	1.08e-2		
	Par.	3.36	-	-	20.6	5.64	3.29	-	8.95	14.3

can be more advantageous if the CPU has more cores (like an eight-core CPU or more).

3.7.2 Standard PGO Benchmarks

Statistics of Each Part

We report the computational time for Cholesky factorization, MCB, and Chordal initialization in Table 3.3. The timings for linearization, state updates, and system matrix constructions are ignored, because these operations are rather cheap compared with the Cholesky part. An exception is CB-PGO using FCB (CB-FCB): The system matrix allocation in this case can be rather expensive by indexing non-zero elements in Jacobian. In case of Manhattan, it takes almost 1 minute to construct the system matrix, and fails in case of City10k and Torus10k. The size and sparsity of the benchmarks are included as well.

Regarding Cholesky factorization, CB-MCB and VB have comparable performance, while CB-MCB is (2-3 times) faster on sparse graphs. The Cholesky part of CB-FCB takes around two magnitudes of time compared with that of CB-MCB/VB, except for the MITb dataset (which has a rather small $\nu = 20$ thus it actually does not matter whether the system matrix is sparse or not). The timing statistics of the Cholesky part indicates that CB-FCB is not a viable approach (2 magnitudes slower than VB in almost any cases).

In Table 3.3, the timing of the MCB and Cholesky parts are comparable for sparse graphs, i.e., the four real dataset, MITb, INTEL_P, KITTI, Intel. Given that we only need to run MCB once, followed by multiple Cholesky iterations, CB-MCB can take advantage in this case, in particular for the MITb, INTEL_P and KITTI datasets. For dense (simulated) graphs, like Sphere, Manhattan, City10k and Torus10k, since the MCB part is too expensive compared with the Cholesky part, it is not a good choice to use CB-MCB (or CB-MCB-Chord) if timing is a critical consideration.

The chordal initialization accounts for roughly 2-3 iterations of the Cholesky time. This is understandable since its solves a linear system of k times larger (with $k = 2$ for 2D; and

Table 3.3 Computational Time of Chordal Initialization, MCB, and Cholesky Factorization on Benchmark Datasets. $\mathcal{G}(\mathcal{V}, \mathcal{E})$ stands for the original graph, while $\bar{\mathcal{G}}(\bar{\mathcal{V}}, \bar{\mathcal{E}})$ the reduced weighted graph. Time in Seconds.

	$\mathcal{G}$				$\bar{\mathcal{G}}$		Chord.	MCB	Cholesky Per Iter.		
	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{V}\rvert$	ν	$\nu/\lvert\mathcal{E}\rvert$	$\lvert\bar{\mathcal{E}}\rvert$	$\lvert\bar{\mathcal{V}}\rvert$	all	CB-MCB	CB-MCB	CB-FCB	VB
MITb	827	808	20	**2.42%**	60	41	7.32e-3	8.45e-4	6.74e-5	6.12e-5	1.31e-3
INTEL_P	1483	1228	256	**17.3%**	396	141	1.18e-2	2.11e-3	1.04e-3	3.98e-2	2.28e-3
KITTI	5065	4541	525	**10.4%**	654	130	5.98e-2	3.94e-3	2.60e-3	1.43	1.02e-2
Intel	1835	943	893	48.7%	1515	623	1.45e-2	3.36e-2	3.31e-3	0.985	2.75e-3
Manhattan	5453	3500	1954	35.8%	4350	2397	4.43e-2	0.348	8.16e-3	1.31	8.42e-3
Sphere2500	4949	2500	2450	49.5%	4947	2498	0.131	0.883	8.51e-2	0.262	8.20e-2
City10k	20687	10000	10688	51.7%	19528	8841	0.209	7.26	6.78e-2	fail	6.11e-2
Torus10k	22280	10000	12281	55.1%	21542	9262	0.589	8.95	0.319	fail	0.340

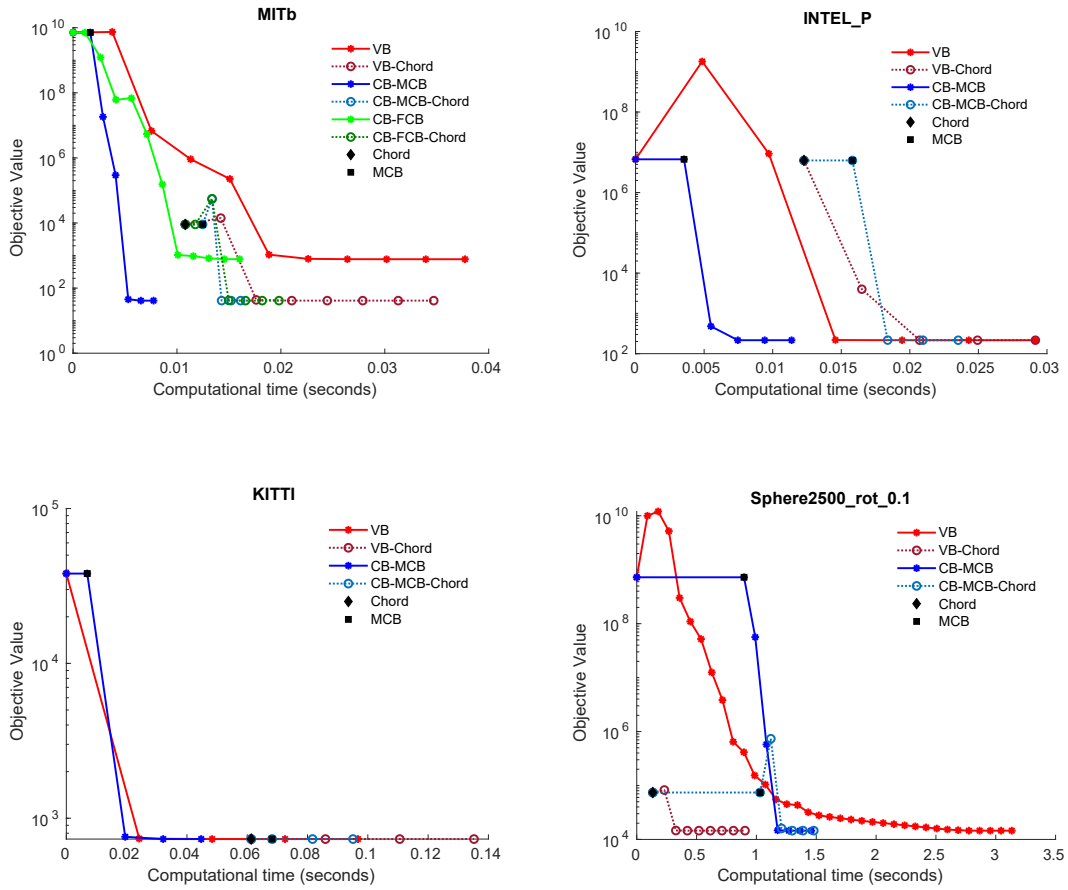


Figure 3.5 The convergence of different methods on benchmark datasets. Noet that for INTEL_P, the initial objective values with and without the chordal initialization are 6281560 and 6700310 respectively, which are close but not the same.

$k = 3$ for 3D). We will use chordal initialization to boost all the methods (termed VB-Chord, CB-MCB-Chord, CB-FCB-Chord) to examine the robustness.

Convergence and Complexity

In Fig. 3.5, we visualize the convergence of VB, VB-Chord, CB-MCB, CB-MCB-Chord, CB-FCB, and CB-FCB-Chord, against their computational time. The time point of the MCB and Chordal initialization are marked out for a clear visualization.

The MTIb dataset shows a case where VB and CB-FCB converges to a local minimum, where CB-MCB converges to the global minimum. The chordal bootstrapped approaches, VB-Chord and CB-FCB-Chord, can converge to the global minimum as well, while taking

a longer time. In this case, CB-MCB is both the fastest and robustest approach. The fast convergence of CB-MCB is further validated by a sparse graph, INTEL_P (17.3% sparsity), where it significantly outperforms VB, and VB-Chord. The results for FCB, FCB-Chord are not shown for INTEL_P as the the timings are too large to be plotted on the same scale. Similar results are witnessed on the KITTI dataset, while the convergence is much faster given the dataset is less noisy. For dense graphs, CB-MCB is not advantageous in terms of the computational time, since the MCB part is dominant over the Cholesky part as shown in Table 3.3. However, even in these cases, CB-MCB is still useful because it yields a faster convergence. For instance on the Sphere2500 dataset, if we increase the noise level to 0.1 rads in the rotation part, the convergence of VB degenerates dramatically, and CB-MCB can outperform VB because of the faster convergence. Nonetheless, the VB-Chord can significantly improve the convergence of VB, thus yielding a smaller timing.

Aside from the MCB, the memory usage of VB and CB are comparable. For example in INTEL_P, the graph storage takes 1.13MB, while the matrix to be factorized accounts for 8.11MB; These numbers in CB are 1.34MB and 8.04MB respectively. For MITb, there are 92 nonzero blocks in the Cholesky part of CB-MCB, whereas 2462 for VB. For INTEL_P, CB-MCB accounts for 2234 nonzero blocks, and VB 4194. The numbers for KITTI are 5015 and 13831 respectively. For dense graphs like Sphere2500, there are 12244 nonzero blocks in the Cholesky part of CB-MCB, while the number for VB is 12398.

3.7.3 Monte-Carlo Simulation

Global Minimum

To examine the robustness of CB-PGO, in terms of converging to the global minimum, we provide a Monte-Carlo simulation on the simulated dataset Manhattan (2D), and Sphere2500 (3D).

We recreate the dataset from its ground-truth with additive noise in the exponential

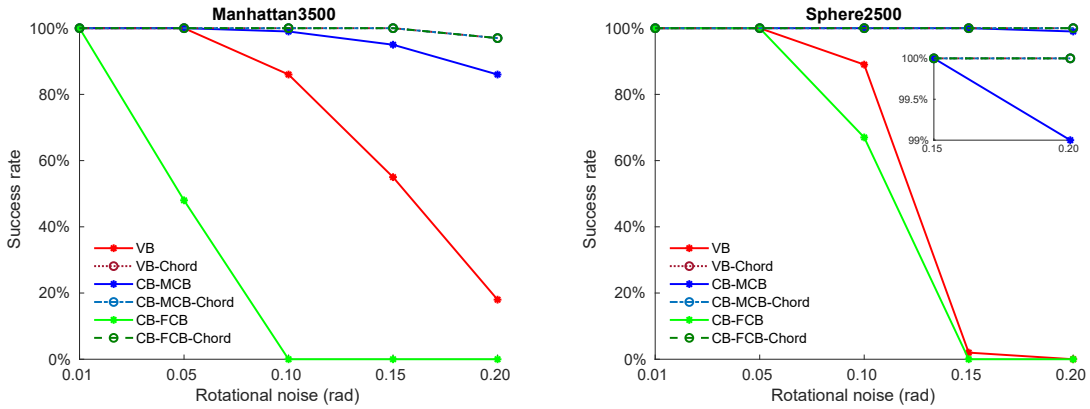


Figure 3.6 The robustness of vertex-based approaches and cycle based approaches on 100 run Monte-Carlo simulation. The curves of the chordal bootstrapped methods, i.e., VB-Chord, CB-MCB-Chord, and CB-FCB-Chord, coincide with one other.

coordinate, with 0.1m standard deviation on the translational part, and 0.01rads 0.05rads, 0.10rads, 0.15rads, 0.20rads respectively on the rotational part.

For each case, we generate 100 noisy graphs, and report in Fig. 3.6 the success rate of VB, VB-Chord, CB-MCB, CB-MCB-Chord, CB-FCB, and CB-FCB-Chord, when solving these graphs. From Fig. 3.6, we conclude that CB-FCB is basically worse than VB, while CB-MCB is much more robust than VB (initialized by odometry) and CB-FCB. Unsurprisingly, the chordal bootstrapped approaches, i.e., VB-Chord, CB-MCB-Chord and CB-FCB-Chord show the best robustness. However, a pure CB-MCB (initialized by relative measurements), is almost as robust as the chordal bootstrapped approaches.

Computational Time

To better understand the computational complexity of VB, VB-Chord, CB-MCB and CB-MCB-Chord on sparse graphs, we perform a Monte-Carlo simulation, with respect to different cycle ratios and different number of poses. Cases for CB-FCB, CB-FCB-Chord are ignored because the FCB based approaches do not scale well with respect to graph topologies (see Table 3.3).

We firstly simulate a giant dense graph using the g2o simulator [105] (g2o_simulator3d).

Then the graph is tailored respectively to 1000, 4000, 10000 and 30000 poses, and pruned to different sparsity, i.e., 1%, 5%, 10%, 15%, 20% and 25%. The process randomly generates 100 graphs for each pose number and cycle ratio combination.

The running time statistics of VB, VB-Chord, CB-MCB and CB-MCB-Chord for each simulated scenario is recorded in Fig. 3.7, where the mean is plotted with color, and the standard deviation is added to the bar-plot as the white margin. The used iterations are also included above each bar. Note that the maximal iteration is set to 50.

In Fig. 3.7, CB-MCB is obviously more advantageous than VB for all the tested scenarios with a maximal cycle ratio at around 15%, as well as VB-Chord at round 10%. When the graph becomes denser, with the cycle ratio exceeding 20%, the timings of CB-MCB and CB-MCB-Chord grow with respect to the number of poses as their MCB parts start to dominate the complexity. However, even in these cases, the timings of CB-MCB and CB-MCB-Chord are still comparable to that of VB and VB-Chord, without a dramatical deterioration. The VB-Chord can improve the convergence of VB with a slight trade-off on the computational cost. The same applies to CB-MCB and CB-MCB-Chord, while the improvement is less obvious. Considering CB-MCB is almost as robust as chordal based approaches, as shown in Fig. 3.6, it seems that CB-MCB-Chord does not offer much compared to CB-MCB.

In case of huge graphs, like 30000 poses, the numerical stability of VB degenerates dramatically, as can be clearly seen by the iterations consumed. In contrast, the CB-MCB shows a much better numerically stability and convergence property in this scenario, while the bootstrapped approach VB-Chord can significantly improve the convergence of VB. It is worth noting that CB-MCB and CB-MCB-Chord can fail when allocating the memory for shortest path trees (a dense matrix of $O(n^2)$ memory) if the reduced graph still has too many vertices. Such a case is shown at 25% sparsity, with 30000 poses.

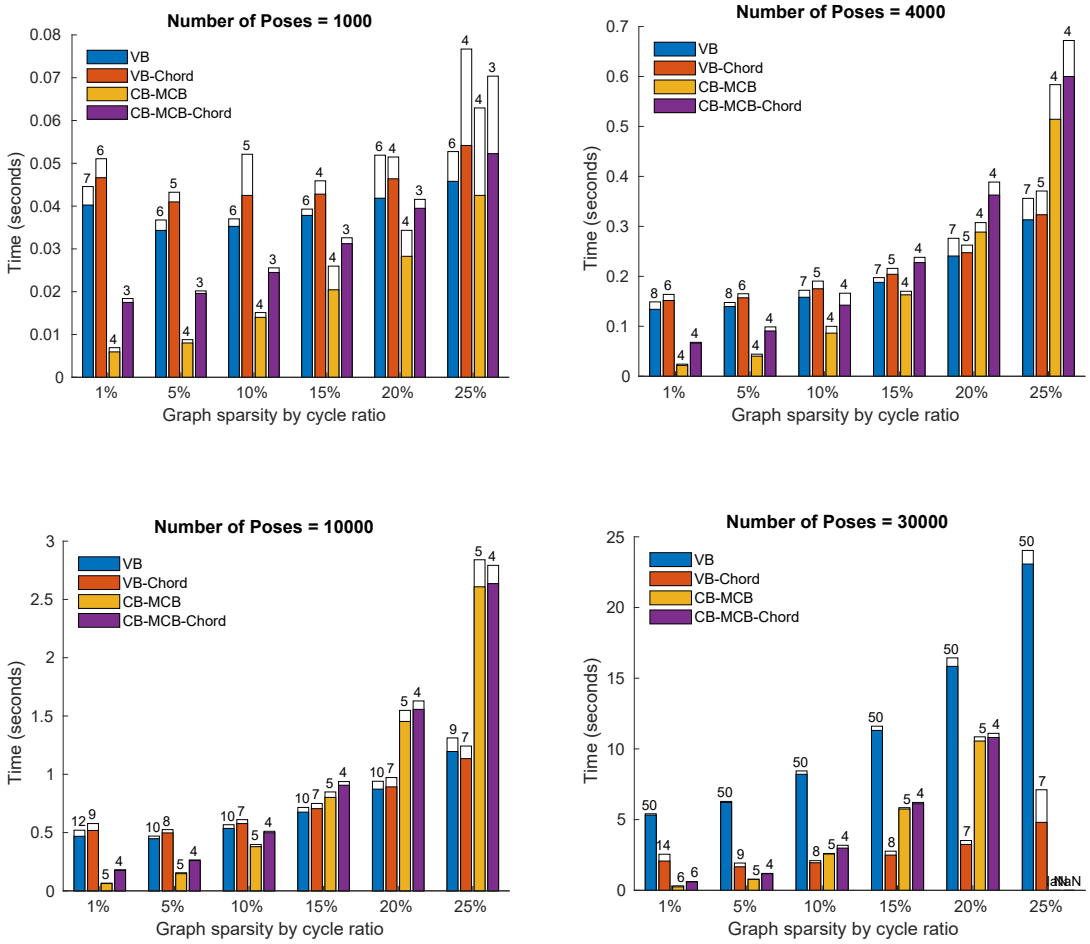


Figure 3.7 The computational time of VB (VB-PGO initialized by odometry), VB-Chord (VB-PGO initialized by the Chordal initialization technique), CB-MCB (CB-PGO based on the MCB, initialized by the measurements of relative poses), and CB-MCB-Chord (CB-PGO based on the MCB, initialized by the Chordal initialization technique) on 100 run Monte-Carlo simulation. The mean is plotted with color, and the standard deviation is added to the bar-plot as the white margin. The number above each bar is the used iterations. The maximal iteration is set to 50.

Chapter Four

Change of Optimal Values

The change of optimal values provides an influence measure in regression diagnostics [67], which is used as a posterior metric for robust estimation designs. See [75, 106] for an implementation of this idea in SLAM. It was thought a common sense that the optimal value of a cost function can only be obtained after solving the problem, therefore methods developed in this line inevitably bare a high computational complexity due to the expensive solving bottleneck. In this chapter, we will break this common sense and show that the change of optimal values can actually be pre-calculated.

Given that GO can be formulated both in the cut space (conventionally) and in the cycle space (developed in Chapter 3), we provide the derivation for both formulations. Mathematically, GO in the cut space takes the form of a least squares optimization; while its counterpart in the cycle space is a minimum norm optimization. Therefore, in the following, we use the terminology "least squares optimization" and "minimum norm optimization". Nonetheless, we can always go back to the notation of cut space and cycle space, if the sparsity of GO is required to enable efficient numerical calculation.

In what follows, we will firstly derive the exact result in the linear case, and provide a trivial extension to the weighted linear variant which is also exact. Then we show how to extend the linear result to nonlinear scenarios based on linearization, particularly providing the extension on manifold. At last, we provide discussions on the commonly encountered

problems when implementing the metric, as well as the connection to information gain.

4.1 Derivation on Linear Least Squares

4.1.1 Problem Statement and Standard Form

Let us consider two incrementally constructed linear least squares optimization problems

$$f^\star = \min \|\mathbf{A}_1\mathbf{x} - \mathbf{b}_1\|^2 \tag{4.1}$$

and

$$f^{\star\star} = \min \|\mathbf{A}_1\mathbf{x} - \mathbf{b}_1\|^2 + \|\mathbf{A}_2\mathbf{x} - \mathbf{b}_2\|^2 = \min \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \tag{4.2}$$

where $\mathbf{A}_1$, $\mathbf{A}_2$ are two "tall" matrices, and we denote

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

Let us further assume $\mathbf{A}_1$ has full column rank.

At some point, we have completely solved the problem (4.1), and obtained the solution $\mathbf{x}^\star$, and its covariance $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$. Then the optimal value reaches $f^\star = \|\mathbf{A}_1\mathbf{x}^\star - \mathbf{b}_1\|^2$. Now the task is that: Given all these information, can we tell something about the optimal value $f^{\star\star}$ without solving the problem (4.2)? Obviously, an equivalent task is to get the change of optimal values $\Delta f = f^{\star\star} - f^\star$.

4.1.2 Classical Results and Preliminaries

To begin with, let us recall some classical results on linear least squares, taking problem (4.2) as an example.

The optimal solution $\mathbf{x}_{opt}$ is

$$\mathbf{x}_{opt} = \mathbf{A}^\dagger\mathbf{b} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}.$$

The covariance (if the underlying noise is modeled as Gaussian) of the optimal solution writes

$$\mathbb{C}\text{ov}(\mathbf{x}_{opt}) = (\mathbf{A}^T\mathbf{A})^{-1}.$$

The optimal value, i.e. the cost at the optimal solution, in this case is $f_{opt} = \|\mathbf{A}\mathbf{x}_{opt} - \mathbf{b}\|^2 = f^{\star\star}$, which can be further written as below by putting $\mathbf{x}_{opt}$ into the cost function:

$$f_{opt} = \mathbf{b}^T[\mathbf{I} - \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T]\mathbf{b} = \mathbf{b}^T(\mathbf{I} - \mathbf{A}\mathbf{A}^\dagger)\mathbf{b}.$$

The Woodbury matrix identity (i.e. the matrix inverse lemma in some references) will be quite useful in later derivation, which states:

Lemma 9 (Woodbury matrix identity)**.**

$$(\mathbf{W} + \mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{W}^{-1} - \mathbf{W}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}\mathbf{W}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{W}^{-1}$$

4.1.3 Three Identities

Let us define two matrices $\mathbf{M}$ and $\mathbf{\Omega}$ as

$$\mathbf{M} = \mathbf{A}^T\mathbf{A} = \mathbf{A}_1^T\mathbf{A}_1 + \mathbf{A}_2^T\mathbf{A}_2$$

$$\mathbf{\Omega} = \mathbf{A}_2(\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_2^T + \mathbf{I} = \mathbf{A}_2\mathbb{C}\text{ov}(\mathbf{x}^\star)\mathbf{A}_2^T + \mathbf{I}.$$

By applying the Woodbury matrix identity to $\mathbf{M}$ and $\mathbf{\Omega}$, we obtain the following two identities:

$$\mathbf{M}^{-1} = (\mathbf{A}_1^T\mathbf{A}_1)^{-1} - (\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_2^T\mathbf{\Omega}^{-1}\mathbf{A}_2(\mathbf{A}_1^T\mathbf{A}_1)^{-1} \tag{4.3}$$

$$\mathbf{\Omega}^{-1} = \mathbf{I} - \mathbf{A}_2\mathbf{M}^{-1}\mathbf{A}_2^T. \tag{4.4}$$

Noting that $\mathbf{A}_1^\dagger = (\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_1^T$ ($\mathbf{A}_1$ has full column rank), from (4.3) it is straightforward to see

$$\mathbf{A}_1\mathbf{M}^{-1}\mathbf{A}_1^T = \mathbf{A}_1\mathbf{A}_1^\dagger - (\mathbf{A}_2\mathbf{A}_1^\dagger)^T\mathbf{\Omega}^{-1}\mathbf{A}_2\mathbf{A}_1^\dagger. \tag{4.5}$$

In addition to Eq. (4.4) and Eq. (4.5), another useful identity regarding $\mathbf{M}$ and $\mathbf{\Omega}$ is

$$\mathbf{A}_2\mathbf{M}^{-1}\mathbf{A}_1^T = \mathbf{\Omega}^{-1}\mathbf{A}_2(\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_1^T = \mathbf{\Omega}^{-1}\mathbf{A}_2\mathbf{A}_1^\dagger. \tag{4.6}$$

Proof. To show the correctness of (4.6), let us start with the trivial identity on $\mathbf{M}$

$$\mathbf{M}(\mathbf{A}_1^T\mathbf{A}_1)^{-1} = \mathbf{I} + (\mathbf{A}_2^T\mathbf{A}_2)(\mathbf{A}_1^T\mathbf{A}_1)^{-1}.$$

Then left-multiply the above equality by $\mathbf{A}_2\mathbf{M}^{-1}$ to reach

$$\mathbf{A}_2(\mathbf{A}_1^T\mathbf{A}_1)^{-1} = \mathbf{A}_2\mathbf{M}^{-1} + \mathbf{A}_2\mathbf{M}^{-1}(\mathbf{A}_2^T\mathbf{A}_2)(\mathbf{A}_1^T\mathbf{A}_1)^{-1}.$$

By some matrix manipulations, we conclude

$$\begin{aligned}\mathbf{A}_2\mathbf{M}^{-1} &= \mathbf{A}_2(\mathbf{A}_1^T\mathbf{A}_1)^{-1} - \mathbf{A}_2\mathbf{M}^{-1}(\mathbf{A}_2^T\mathbf{A}_2)(\mathbf{A}_1^T\mathbf{A}_1)^{-1} \\ &\xlongequal{\text{reorganize terms}} [\mathbf{I} - \mathbf{A}_2\mathbf{M}^{-1}\mathbf{A}_2^T]\mathbf{A}_2(\mathbf{A}_1^T\mathbf{A}_1)^{-1} \\ &\xlongequal{\text{apply Eq. (4.4)}} \mathbf{\Omega}^{-1}\mathbf{A}_2(\mathbf{A}_1^T\mathbf{A}_1)^{-1}.\end{aligned}$$

At last, right-multiply by $\mathbf{A}_1^T$, then we will reach (4.6). □

The identities (4.4) (4.5) (4.6) regarding $\mathbf{M}$ and $\mathbf{\Omega}$ are essential for the later proof.

4.1.4 Derivation of the Main Result

The optimal value of the problem (4.1), i.e., $f^\star$, writes

$$f^\star = \mathbf{b}_1^T(\mathbf{I} - \mathbf{A}_1\mathbf{A}_1^\dagger)\mathbf{b}_1 = \mathbf{b}^T\left[\mathbf{I} - \begin{bmatrix}\mathbf{A}_1\mathbf{A}_1^\dagger & \mathbf{O} \\ \mathbf{O} & \mathbf{I}\end{bmatrix}\right]\mathbf{b}.$$

Note that $\mathbf{M} = \mathbf{A}^T\mathbf{A}$. By splitting $\mathbf{A}$ into $\mathbf{A}_1$ and $\mathbf{A}_2$, we can expand the optimal value of the problem (4.2), i.e., $f^{\star\star}$, as

$$\begin{aligned}f^{\star\star} &= \mathbf{b}^T[\mathbf{I} - \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T]\mathbf{b} \\ &= \mathbf{b}^T\left[\mathbf{I} - \begin{bmatrix}\mathbf{A}_1\mathbf{M}^{-1}\mathbf{A}_1^T & \mathbf{A}_1\mathbf{M}^{-1}\mathbf{A}_2^T \\ \mathbf{A}_2\mathbf{M}^{-1}\mathbf{A}_1^T & \mathbf{A}_2\mathbf{M}^{-1}\mathbf{A}_2^T\end{bmatrix}\right]\mathbf{b}.\end{aligned}$$

Now let us examine the change of optimal values $\Delta f = f^{\star\star} - f^{\star}$ from the problem (4.1) to the problem (4.2):

$$
\begin{aligned}
&\Delta f = f^{\star\star} - f^{\star} \\
&= \mathbf{b}^T \begin{bmatrix} \mathbf{A}_1\mathbf{A}_1^\dagger - \mathbf{A}_1\mathbf{M}^{-1}\mathbf{A}_1^T & -\mathbf{A}_1\mathbf{M}^{-1}\mathbf{A}_2^T \\ -\mathbf{A}_2\mathbf{M}^{-1}\mathbf{A}_1^T & \mathbf{I} - \mathbf{A}_2\mathbf{M}^{-1}\mathbf{A}_2^T \end{bmatrix} \mathbf{b} \\
&\xlongequal{\text{by (4.4)(4.5)(4.6)}} \mathbf{b}^T \begin{bmatrix} (\mathbf{A}_2\mathbf{A}_1^\dagger)^T\boldsymbol{\Omega}^{-1}\mathbf{A}_2\mathbf{A}_1^\dagger & -(\mathbf{A}_2\mathbf{A}_1^\dagger)^T\boldsymbol{\Omega}^{-1} \\ -\boldsymbol{\Omega}^{-1}\mathbf{A}_2\mathbf{A}_1^\dagger & \boldsymbol{\Omega}^{-1} \end{bmatrix} \mathbf{b} \\
&= \mathbf{b}^T \cdot \begin{bmatrix} (\mathbf{A_2}\mathbf{A}_1^\dagger)^T \\ -\mathbf{I} \end{bmatrix} \cdot \boldsymbol{\Omega}^{-1} \cdot \begin{bmatrix} \mathbf{A_2}\mathbf{A}_1^\dagger & -\mathbf{I} \end{bmatrix} \cdot \mathbf{b} \\
&= \begin{bmatrix} \mathbf{A_2}\mathbf{A}_1^\dagger\mathbf{b}_1 - \mathbf{b}_2 \end{bmatrix}^T \cdot \boldsymbol{\Omega}^{-1} \cdot \begin{bmatrix} \mathbf{A_2}\mathbf{A}_1^\dagger\mathbf{b}_1 - \mathbf{b}_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A_2}\mathbf{x}^\star - \mathbf{b}_2 \end{bmatrix}^T \cdot [\mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T + \mathbf{I}]^{-1} \cdot \begin{bmatrix} \mathbf{A_2}\mathbf{x}^\star - \mathbf{b}_2 \end{bmatrix}
\end{aligned}
$$

Conclusion: The change of optimal values $\Delta f = f^{\star\star} - f^{\star}$ from the problem (4.1) to the problem (4.2) is

$$\Delta f = (\mathbf{A_2}\mathbf{x}^\star - \mathbf{b}_2)^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T + \mathbf{I}]^{-1}(\mathbf{A_2}\mathbf{x}^\star - \mathbf{b}_2). \tag{4.7}$$

In (4.7), both $\mathbf{x}^\star$ and $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$ have been solved from the previous problem (4.1). Therefore given the new measurement $\mathbf{A}_2$ and $\mathbf{b}_2$, Δf can be pre-calculated in closed form without solving the problem (4.2).

4.2 Derivation on Linear Minimum Norm Optimization

4.2.1 Problem Statement and Standard Form

Let us consider a minimum norm optimization problem

$$\min\ \mathbf{x}^T\mathbf{x} \qquad \mathbf{s.t.}\quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.8}$$

where the constraints are classified into two parts:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

This is particularly the case for temporally incremental optimization problems. At a certain point, the constraint $\mathbf{A}_1\mathbf{x} = \mathbf{b}_1$ represents those that are already available at a specific time point, while the constraint $\mathbf{A}_2\mathbf{x} = \mathbf{b}_2$ can only be obtained after that time point.

The optimization problem established using the previously available measurement explicitly writes

$$\min \ \mathbf{x}^T\mathbf{x} \qquad \mathbf{s.t.} \quad \mathbf{A}_1\mathbf{x} = \mathbf{b}_1. \tag{4.9}$$

Now at some point, we have solved the problem (4.9), but not yet problem (4.8). Let us denote the solution of the problem (4.9) as $\mathbf{x}^\star$ and the corresponding covariance matrix as $\mathbb{C}\text{ov}(\mathbf{x}^\star)$. The question is that: Can we tell something useful about problem (4.8) from the results of problem (4.9)?

In this section, we establish the equation on how to calculate the optimal value (of the objective function) of problem (4.8), without solving it directly.

4.2.2 Classical Results and Preliminaries

To proceed, let us recall some classical results of minimum norm optimization, taking problem (4.8) as an example.

The optimal solution $\mathbf{x}_{opt}$ is

$$\mathbf{x}_{opt} = \mathbf{A}^\dagger\mathbf{b} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}.$$

The covariance (with Gaussian assumption) of the optimal solution, $\mathbb{C}\text{ov}(\mathbf{x}_{opt})$, is the projection matrix to the Null space of $\mathbf{A}$, which writes

$$\mathbb{C}\text{ov}(\mathbf{x}_{opt}) = \mathbf{P}_{(N(\mathbf{A}))} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}.$$

The cost at the optimal solution, termed the optimal value (of the objective function), i.e. $\mathbf{x}_{opt}^T\mathbf{x}_{opt}$, writes

$$f_{opt} = \mathbf{b}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}.$$

We will use the Schur complement [120] to decompose the below block matrix

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{O} \\ \mathbf{O} & \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}. \tag{4.10}$$

Eq. (4.10) stands as long as $\mathbf{A}$ is invertible.

4.2.3 Derivation of the Main Result

The derivation consists of a manipulation of matrices via linear algebra laws. Basically, we will expand the optimal value of problem (4.8), i.e., $\mathbf{b}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}$, and examine its components.

The two matrices below are useful to express the Schur complement of $\mathbf{A}\mathbf{A}^T$. Let us define

$$\mathbf{W} = \mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T = \mathbf{A}_2[\mathbf{I} - \mathbf{A}_1^T(\mathbf{A}_1\mathbf{A}_1^T)^{-1}\mathbf{A}_1]\mathbf{A}_2^T \tag{4.11}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{A}_2\mathbf{A}_1^T(\mathbf{A}_1\mathbf{A}_1^T)^{-1} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{A}_2\mathbf{A}_1^\dagger & \mathbf{I} \end{bmatrix} \tag{4.12}$$

Let us apply Schur complement (4.10) to the matrix $\mathbf{A}\mathbf{A}^T$. The result writes

$$\begin{aligned} \mathbf{A}\mathbf{A}^T &= \begin{bmatrix} \mathbf{A}_1\mathbf{A}_1^T & \mathbf{A}_1\mathbf{A}_2^T \\ \mathbf{A}_2\mathbf{A}_1^T & \mathbf{A}_2\mathbf{A}_2^T \end{bmatrix} \\ &= \mathbf{H}\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1^T & \mathbf{O} \\ \mathbf{O} & \mathbf{A}_2\mathbf{A}_2^T - \mathbf{A}_2\mathbf{A}_1^T(\mathbf{A}_1\mathbf{A}_1^T)^{-1}\mathbf{A}_1\mathbf{A}_2^T \end{bmatrix}\mathbf{H}^T \\ &= \mathbf{H}\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1^T & \mathbf{O} \\ \mathbf{O} & \mathbf{W} \end{bmatrix}\mathbf{H}^T. \end{aligned}$$

Noting that $\mathbf{H}^{-1}\mathbf{b}$ can be written as

$$\mathbf{H}^{-1}\mathbf{b} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ -\mathbf{A}_2\mathbf{A}_1^\dagger & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ -\mathbf{A}_2\mathbf{A}_1^\dagger\mathbf{b}_1 + \mathbf{b}_2 \end{bmatrix},$$

then we can finally expand $\mathbf{b}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}$ by a straightforward matrix calculation as follow

$$\begin{aligned}
&f^{\star\star} = \mathbf{b}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \\
=&\mathbf{b}^T\mathbf{H}^{-T}\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1^T & \mathbf{O} \\ \mathbf{O} & \mathbf{W} \end{bmatrix}^{-1}\mathbf{H}^{-1}\mathbf{b} \\
=&(\mathbf{H}^{-1}\mathbf{b})^T\begin{bmatrix} (\mathbf{A}_1\mathbf{A}_1^T)^{-1} & \mathbf{O} \\ \mathbf{O} & \mathbf{W}^{-1} \end{bmatrix}\mathbf{H}^{-1}\mathbf{b} \\
=&\begin{bmatrix} \mathbf{b}_1 \\ -\mathbf{A}_2\mathbf{A}_1^\dagger\mathbf{b}_1 + \mathbf{b}_2 \end{bmatrix}^T\begin{bmatrix} (\mathbf{A}_1\mathbf{A}_1^T)^{-1} & \mathbf{O} \\ \mathbf{O} & \mathbf{W}^{-1} \end{bmatrix}\begin{bmatrix} \mathbf{b}_1 \\ -\mathbf{A}_2\mathbf{A}_1^\dagger\mathbf{b}_1 + \mathbf{b}_2 \end{bmatrix} \\
=&\mathbf{b}_1^T(\mathbf{A}_1\mathbf{A}_1^T)^{-1}\mathbf{b}_1 + (\mathbf{A}_2\mathbf{A}_1^\dagger\mathbf{b}_1 - \mathbf{b}_2)^T\mathbf{W}^{-1}(\mathbf{A}_2\mathbf{A}_1^\dagger\mathbf{b}_1 - \mathbf{b}_2) \\
=&f^\star + \underbrace{(\mathbf{A}_2\mathbf{x}^\star - \mathbf{b}_2)^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T]^{-1}(\mathbf{A}_2\mathbf{x}^\star - \mathbf{b}_2)}_{\Delta f}.
\end{aligned}$$

Conclusion: Therefore the **change of optimal values**, $\Delta f = f^{\star\star} - f^\star$, from problem (4.8) to problem (4.9) can be compactly written as

$$\Delta f = (\mathbf{A}_2\mathbf{x}^\star - \mathbf{b}_2)^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T]^{-1}(\mathbf{A}_2\mathbf{x}^\star - \mathbf{b}_2). \tag{4.13}$$

In (4.13), the unknown piece of information, $\mathbf{x}^\star$ and $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$, can be obtained by solving the problem (4.9). In other words, given $\mathbf{x}^\star$ and its covariance $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$, we can calculate Δf through Eq. (4.13), then we calculate the the optimal value of problem (4.8) simply by $f^{\star\star} = f^\star + \Delta f$, without solving it!

4.3 Extension to Linear Variants

4.3.1 Extension to Weighted Linear Least Squares

In this section, we will extend the result in (4.7) to weighted least squares. Let us consider two sequential constructed weighted least squares optimization problems

$$\min\ (\mathbf{H}_1\mathbf{x}-\mathbf{h}_1)^T\mathbf{\Sigma}_1^{-1}(\mathbf{H}_1\mathbf{x}-\mathbf{h}_1) = \min\ \|\bar{\mathbf{H}}_1\mathbf{x}-\bar{\mathbf{h}}_1\|^2 \tag{4.14}$$

$$\min\ (\mathbf{H}\mathbf{x}-\mathbf{h})^T\mathbf{\Sigma}^{-1}(\mathbf{H}\mathbf{x}-\mathbf{h}) = \min\ \|\bar{\mathbf{H}}\mathbf{x}-\bar{\mathbf{h}}\|^2 \tag{4.15}$$

where the involved matrices are

$$\mathbf{H}=\begin{bmatrix}\mathbf{H}_1\\ \mathbf{H}_2\end{bmatrix},\quad \mathbf{h}=\begin{bmatrix}\mathbf{h}_1\\ \mathbf{h}_2\end{bmatrix},\quad \mathbf{\Sigma}=\begin{bmatrix}\mathbf{\Sigma}_1 & \\ & \mathbf{\Sigma}_2\end{bmatrix}.$$

As before, we assume both $\mathbf{H}_1$ and $\mathbf{H}_2$ are "tall" matrices, and have full column rank. $\mathbf{\Sigma}$ is positive definite.

Here in (4.14)(4.15), we convert the problem to the standard form, i.e., (4.1)(4.2), by absorbing the weight into the coefficient. In specific, we let $\bar{\mathbf{H}}_1 = \mathbf{\Sigma}_1^{-\frac{1}{2}}\mathbf{H}_1$, $\bar{\mathbf{h}}_1 = \mathbf{\Sigma}_1^{-\frac{1}{2}}\mathbf{h}_1$, $\bar{\mathbf{H}}_2 = \mathbf{\Sigma}_2^{-\frac{1}{2}}\mathbf{H}_2$, $\bar{\mathbf{h}}_2 = \mathbf{\Sigma}_2^{-\frac{1}{2}}\mathbf{h}_2$, which implies $\bar{\mathbf{H}} = \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{H}$, $\bar{\mathbf{h}} = \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{h}$.

The optimal solution of the problem (4.14) and the corresponding covariance matrix write

$$\mathbf{x}^\star = (\bar{\mathbf{H}}_1^T\bar{\mathbf{H}}_1)^{-1}\bar{\mathbf{H}}_1^T\bar{\mathbf{h}}_1 = (\mathbf{H}_1^T\mathbf{\Sigma}_1^{-1}\mathbf{H}_1)^{-1}\mathbf{H}_1^T\mathbf{\Sigma}_1^{-1}\mathbf{h}_1$$

$$\mathbb{C}\mathrm{ov}(\mathbf{x}^\star) = (\bar{\mathbf{H}}_1^T\bar{\mathbf{H}}_1)^{-1} = (\mathbf{H}_1^T\mathbf{\Sigma}_1^{-1}\mathbf{H}_1)^{-1}.$$

A straightforward matrix calculation shows that

$$\bar{\mathbf{H}}_2\mathbf{x}^\star - \bar{\mathbf{h}}_2 = \mathbf{\Sigma}_2^{-\frac{1}{2}}(\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2)$$

$$[\bar{\mathbf{H}}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\bar{\mathbf{H}}_2^T + \mathbf{I}]^{-1} = \mathbf{\Sigma}_2^{\frac{1}{2}}[\mathbf{H}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{H}_2^T + \mathbf{\Sigma}_2]^{-1}\mathbf{\Sigma}_2^{\frac{1}{2}}.$$

Then by the result in (4.7), we have

$$\begin{aligned}\Delta f &= (\bar{\mathbf{H}}_2\mathbf{x}^\star - \bar{\mathbf{h}}_2)^T[\bar{\mathbf{H}}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\bar{\mathbf{H}}_2^T + \mathbf{I}]^{-1}(\bar{\mathbf{H}}_2\mathbf{x}^\star - \bar{\mathbf{h}}_2)\\ &= (\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2)^T[\mathbf{H}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{H}_2^T + \mathbf{\Sigma}_2]^{-1}(\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2).\end{aligned}$$

Conclusion: The change of optimal values from the problem (4.14) to the problem (4.15) is

$$\Delta f = (\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2)^T[\mathbf{H}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{H}_2^T + \mathbf{\Sigma}_2]^{-1}(\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2). \tag{4.16}$$

4.3.2 Extension to Linear Least Distance Optimization

In this section, we consider the problem of a least distance optimization, whose standard form writes

$$\min\ (\mathbf{H}\mathbf{x} - \mathbf{h})^T\mathbf{\Sigma}^{-1}(\mathbf{H}\mathbf{x} - \mathbf{h}) \qquad \mathbf{s.t.}\quad \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{4.17}$$

In (4.17), we assume $\mathbf{H}$ to be invertible. In this case, by letting $\mathbf{y} = \mathbf{\Sigma}^{-\frac{1}{2}}(\mathbf{H}\mathbf{x} - \mathbf{h})$, i.e. $\mathbf{x} = \mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{y} + \mathbf{H}^{-1}\mathbf{h}$, (4.17) can be converted to a least norm optimization

$$\min\ \mathbf{y}^T\mathbf{y} \qquad \mathbf{s.t.}\quad \mathbf{A}\mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{y} = \mathbf{b} - \mathbf{A}\mathbf{H}^{-1}\mathbf{h}. \tag{4.18}$$

Therefore we can extend the conclusion on the least norm optimization to (4.17).

As before, we assume $\mathbf{A}$ and $\mathbf{b}$ are obtained in two different phases comprising

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

Let us write the optimization problem using the measurement from the first phase only as

$$\min\ (\mathbf{H}\mathbf{x} - \mathbf{h})^T\mathbf{\Sigma}^{-1}(\mathbf{H}\mathbf{x} - \mathbf{h}) \qquad \mathbf{s.t.}\quad \mathbf{A}_1\mathbf{x} = \mathbf{b}_1. \tag{4.19}$$

Formally, we would like to calculate the change of optimal values from the problem (4.19) to the problem (4.17).

Let us write (4.19) in the form of a least norm optimization

$$\min\ \mathbf{y}^T\mathbf{y} \qquad \mathbf{s.t.}\quad \mathbf{A}_1\mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{y} = \mathbf{b}_1 - \mathbf{A}\mathbf{H}^{-1}\mathbf{h} \tag{4.20}$$

and denote its optimal solution as $\mathbf{y}^\star$. The covariance of $\mathbf{y}^\star$ is related with that of $\mathbf{x}^\star = \mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{y}^\star + \mathbf{H}^{-1}\mathbf{h}$ by the linear transformation of Gaussian distribution

$$\mathbf{A}_2\mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbb{C}\mathrm{ov}(\mathbf{y}^\star)\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{H}^{-T}\mathbf{A}_2^T = \mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T, \tag{4.21}$$

and considering $\mathbf{y}^\star = \mathbf{\Sigma}^{-\frac{1}{2}}(\mathbf{H}\mathbf{x}^\star - \mathbf{h}_1)$ we have

$$\mathbf{A}_2\mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{y}^\star - (\mathbf{b}_2 - \mathbf{A}_2\mathbf{H}^{-1}\mathbf{h}_2) = \mathbf{A}_2\mathbf{x}^\star - \mathbf{b}_2. \tag{4.22}$$

Inserting (4.21) and (4.22) into the result of minimum norm optimization, we immediately conclude that the change of optimal values writes

$$\Delta f = (\mathbf{A}_2\mathbf{x}^\star - \mathbf{b}_2)^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T]^{-1}(\mathbf{A}_2\mathbf{x}^\star - \mathbf{b}_2). \tag{4.23}$$

At last, we can write down a formula of $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$ by considering together

$$\mathbb{C}\mathrm{ov}(\mathbf{y}^\star) = \mathbf{I} - \mathbf{\Sigma}^{\frac{1}{2}}\mathbf{H}^{-T}\mathbf{A}_1^T(\mathbf{A}_1\mathbf{H}^{-1}\mathbf{\Sigma}\mathbf{H}^{-T}\mathbf{A}_1^T)^{-1}\mathbf{A}_1\mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}$$

$$\mathbb{C}\mathrm{ov}(\mathbf{x}^\star) = \mathbf{H}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbb{C}\mathrm{ov}(\mathbf{y}^\star)\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{H}^{-T}$$

to reach its explicit form

$$\mathbb{C}\mathrm{ov}(\mathbf{x}^\star) = \mathbf{Q} - \mathbf{Q}\mathbf{A}_1^T(\mathbf{A}_1\mathbf{Q}\mathbf{A}_1^T)^{-1}\mathbf{A}_1\mathbf{Q} \tag{4.24}$$

where $\mathbf{Q} = \mathbf{H}^{-1}\mathbf{\Sigma}\mathbf{H}^{-T}$.

To conclude, if we have known $\mathbf{x}^\star$ and $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$ by solving the problem (4.19), then we can predict the change of optimal values between (4.19) and (4.17) by (4.23) easily. In the linear case, the result is exact. For nonlinear cases, a linear approximation may apply around its current working point.

4.4 Extension to Nonlinear Cases on Manifold

4.4.1 Extension to Nonlinear Least Squares

Let $\mathbf{x}$ be the (potentially multi-dimensional) state variable, which can be observed by a nonlinear measurement model

$$\mathbf{z} = \mathbf{m}(\mathbf{x}).$$

The model $\mathbf{m}(\cdot)$ can be obtained by analyzing the physical mechanism of the underlying system, or learnt from a bunch of data by regression techniques. The variable $\mathbf{z}$ is in practice obtained from sensor readings, which is typically contaminated by process noise. Let the noisy observation of $\mathbf{z}$ be $\tilde{\mathbf{z}}$.

Given the noisy measurement $\tilde{\mathbf{z}}$, the maximum likelihood estimate (MLE) of $\mathbf{x}$ can be obtained by solving a nonlinear least squares optimization problem

$$\min \|\mathbb{D}\text{iff}(\mathbf{m}(\mathbf{x}), \tilde{\mathbf{z}})\|^2_{\mathbf{\Sigma}},$$

where $\mathbb{D}\text{iff}(\mathbf{m}(\mathbf{x}), \tilde{\mathbf{z}})$ is a vectorized residual error function between $\mathbf{m}(\mathbf{x})$ and $\tilde{\mathbf{z}}$. Σ controls the weight of the residual error, which is usually related to the covariance of a Gaussian probability distribution.

Note that $\mathbf{m}(\mathbf{x})$ and $\mathbf{z}$ would always possess the same dimension. Therefore, it is always possible to define $\mathbb{D}\text{iff}(\cdot, \cdot)$ as a Euclidean difference, i.e., $\mathbb{D}\text{iff}(\mathbf{m}(\mathbf{x}), \tilde{\mathbf{z}}) = \mathbf{m}(\mathbf{x}) - \tilde{\mathbf{z}}$. In what follows, we extend the Euclidean minus "$-$" to the manifold, which is a typical scenario in robotic applications. We assume that $\mathbf{m}(\mathbf{x})$ and $\mathbf{z}$ are two elements on the same manifold, and let us define $\boxminus$ as the difference in the tangent space, i.e., we let $\mathbb{D}\text{iff}(\mathbf{m}(\mathbf{x}), \tilde{\mathbf{z}}) = \mathbf{m}(\mathbf{x}) \boxminus \tilde{\mathbf{z}}$.

Then let us consider two sequentially constructed nonlinear least squares optimization problems:

$$f^\star = \min \|\mathbf{m}_1(\mathbf{x}) \boxminus \tilde{\mathbf{z}}_1\|^2_{\mathbf{\Sigma}_1} \tag{4.25}$$

$$f^{\star\star} = \min \|\mathbf{m}_1(\mathbf{x}) \boxminus \tilde{\mathbf{z}}_1\|^2_{\mathbf{\Sigma}_1} + \|\mathbf{m}_2(\mathbf{x}) \boxminus \tilde{\mathbf{z}}_2\|^2_{\mathbf{\Sigma}_2}. \tag{4.26}$$

As before, our task is to forecast the change of optimal values $\Delta f = f^{\star\star} - f^\star$ between the problem (4.25) and (4.26). In the rest of this section, we show how to obtain an approximation to Δf by using the result in (4.16).

Let us define $\boxplus$ as a retraction on the manifold. $\boxplus$ is used to apply a perturbation $\boldsymbol{\xi}$ in the tangent space of the estimate $\mathbf{x}^\star$, and return $\mathbf{x}^\star \boxplus \boldsymbol{\xi}$ as an element on the manifold. Then the linearized version of the problem (4.25) and (4.26) writes

$$f_l^\star = \min \|\mathbf{h}_1 - \mathbf{H}_1 \boldsymbol{\xi}\|^2_{\mathbf{\Sigma}_1} \tag{4.27}$$

$$f_l^{\star\star} = \min \|\mathbf{h}_1 - \mathbf{H}_1\boldsymbol{\xi}\|_{\boldsymbol{\Sigma}_1}^2 + \|\mathbf{h}_2 - \mathbf{H}_2\boldsymbol{\xi}\|_{\boldsymbol{\Sigma}_2}^2. \tag{4.28}$$

with

$$\mathbf{H}_1 = -\frac{\mathbf{m}_1(\mathbf{x}^\star \boxplus \boldsymbol{\xi}) \boxminus \tilde{\mathbf{z}}_1}{\partial \boldsymbol{\xi}^T}|_{\boldsymbol{\xi}=\mathbf{0}}$$

$$\mathbf{H}_2 = -\frac{\mathbf{m}_2(\mathbf{x}^\star \boxplus \boldsymbol{\xi}) \boxminus \tilde{\mathbf{z}}_2}{\partial \boldsymbol{\xi}^T}|_{\boldsymbol{\xi}=\mathbf{0}}$$

$$\mathbf{h}_1 = \mathbf{m}_1(\mathbf{x}^\star) \boxminus \tilde{\mathbf{z}}_1$$

$$\mathbf{h}_2 = \mathbf{m}_2(\mathbf{x}^\star) \boxminus \tilde{\mathbf{z}}_2.$$

Applying the linear result in (4.16), we have

$$\begin{aligned}
&\Delta f_l = f_l^{\star\star} - f_l^\star \\
&= (\mathbf{H}_2\boldsymbol{\xi}^\star - \mathbf{h}_2)^T[\mathbf{H}_2\mathbb{C}\mathrm{ov}(\boldsymbol{\xi}^\star)\mathbf{H}_2^T + \boldsymbol{\Sigma}_2]^{-1}(\mathbf{H}_2\boldsymbol{\xi}^\star - \mathbf{h}_2) \\
&\xlongequal{\boldsymbol{\xi}^\star=\mathbf{0},\ \mathbb{C}\mathrm{ov}(\boldsymbol{\xi}^\star)=\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)} \mathbf{h}_2^T[\mathbf{H}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{H}_2^T + \boldsymbol{\Sigma}_2]^{-1}\mathbf{h}_2.
\end{aligned}$$

The last equality holds because $\boldsymbol{\xi}^\star = \mathbf{0}$ upon the convergence of $\mathbf{x}^\star$, and the covariance in the tangent space writes $\mathbb{C}\mathrm{ov}(\boldsymbol{\xi}^\star) = (\mathbf{H}_1^T\boldsymbol{\Sigma}_1^{-1}\mathbf{H}_1)^{-1} = \mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$.

In practice, $\Delta f_l = f_l^{\star\star} - f_l^\star$ can be used as an approximation to $\Delta f = f^{\star\star} - f^\star$.

Conclusion: The change of optimal values from the problem (4.25) to the problem (4.26) can be approximated as

$$\Delta f \approx \mathbf{h}_2^T[\mathbf{H}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{H}_2^T + \boldsymbol{\Sigma}_2]^{-1}\mathbf{h}_2. \tag{4.29}$$

4.4.2 Extension to Nonlinear Least Distance Optimization

A nonlinear least distance optimization with equality constraints can be generalized as

$$\min\ \mathbb{D}\mathrm{ist}(\mathbf{x}, \tilde{\mathbf{x}}) \qquad \mathbf{s.t.}\quad \mathbf{C}(\mathbf{x}) = \mathbf{0}$$

with $\tilde{\mathbf{x}}$ being the measurement data having the same dimension as $\mathbf{x}$, and $\mathbb{D}\mathrm{ist}(\mathbf{x}, \tilde{\mathbf{x}})$ being a scalar metric.

Usually, there are many ways to define a distance function $\mathbb{D}\text{ist}(\mathbf{x}, \tilde{\mathbf{x}})$. Here we specifically focus on the Mahalanobis distance, which extends well to the Frobenius norm and ℓ_2 norm based distance. Explicitly, the problem we will discuss takes the form of

$$\min \ \|\mathbb{D}\text{iff}(\mathbf{x}, \tilde{\mathbf{x}})\|_{\Sigma}^2 \qquad \mathbf{s.t.} \quad \mathbf{C}(\mathbf{x}) = \mathbf{0} \tag{4.30}$$

where $\mathbb{D}\text{iff}(\mathbf{x}, \tilde{\mathbf{x}})$ is a vectorized difference between $\mathbf{x}$ and $\tilde{\mathbf{x}}$. Σ models our confidence on $\mathbb{D}\text{iff}(\mathbf{x}, \tilde{\mathbf{x}})$ which is usually interpreted as the covariance of some Gaussian distribution.

A standard extension is to linearize (4.30) to the form of (4.17), and then apply the result in linear cases, as the paradigm found in Extended Kalman Filter (EKF) techniques. In essence, the objective function in the linearized problem is an *approximation* to that of its original nonlinear scenario, which means the result is not exact any more. We mention EKF as an example because (4.23) works incrementally, which shows some similarities to the Kalman Filter (KF). *For an incremental optimization problem, the linearization point does not change too much.* Therefore it is reasonable to assume that the linearized Jacobian provides a valid approximation to the original nonlinear problem.

In what follows, we illustrate the idea of using linearization with an example on manifold.

Let $\mathbf{x}$ and $\tilde{\mathbf{x}}$ be two elements on the same manifold. We use $\boxminus$ to describe their difference in the tangent space, i.e. $\mathbb{D}\text{iff}(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{x} \boxminus \tilde{\mathbf{x}}$. At a certain time point, we have an optimization problem, say **Phase I**:

$$\begin{aligned} &\min \ \|\mathbf{x} \boxminus \tilde{\mathbf{x}}\|_{\Sigma}^2 \\ \mathbf{s.t.} \quad &\mathbf{C}_1(\mathbf{x}) = \mathbf{0} \end{aligned} \qquad (\textbf{Phase I})$$

Let $\mathbf{x}^\star$ and $\mathbb{C}\text{ov}(\mathbf{x}^\star)$ be the solution and covariance of the optimization problem in **Phase I**. Now if we obtain another constraint $\mathbf{C}_2(\mathbf{x}) = \mathbf{0}$, which constitutes another optimization problem named **Phase II**:

$$\begin{aligned} &\min \ \|\mathbf{x} \boxminus \tilde{\mathbf{x}}\|_{\Sigma}^2 \\ \mathbf{s.t.} \quad &\mathbf{C}_1(\mathbf{x}) = \mathbf{0} \\ &\mathbf{C}_2(\mathbf{x}) = \mathbf{0} \end{aligned} \qquad (\textbf{Phase II})$$

Conclusion: Then the change of optimal values from **Phase I** to **Phase II** can be approximated by

$$\Delta f \approx \mathbf{C}_2(\mathbf{x}^\star)^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T]^{-1}\mathbf{C}_2(\mathbf{x}^\star) \tag{4.31}$$

with $\mathbf{A}_2$ given by

$$\mathbf{A}_2 = -\frac{\partial \mathbf{C}_2(\mathbf{x}^\star \boxplus \boldsymbol{\xi})}{\partial \boldsymbol{\xi}^T}|_{\boldsymbol{\xi}=\mathbf{0}}.$$

The notation $\boxplus$ is used to apply a perturbation ξ in the tangent space of $\mathbf{x}^\star$, and return an element on manifold. The covariance $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$ is given in the tangent space of the solution $\mathbf{x}^\star$. It is calculated by by Eq. (4.24), by letting

$$\mathbf{A}_1 = -\frac{\partial \mathbf{C}_1(\mathbf{x}^\star \boxplus \boldsymbol{\xi})}{\partial \boldsymbol{\xi}^T}|_{\boldsymbol{\xi}=\mathbf{0}}$$

$$\mathbf{H} = -\frac{\partial \left((\mathbf{x}^\star \boxplus \boldsymbol{\xi}) \boxminus \tilde{\mathbf{x}}\right)}{\partial \boldsymbol{\xi}^T}|_{\boldsymbol{\xi}=\mathbf{0}}.$$

The derivation of Eq. (4.31) is quite straightforward via linearization. Let

$$\mathbf{h} = \mathbf{x}^\star \boxminus \tilde{\mathbf{x}}, \qquad \mathbf{b}_1 = \mathbf{C}_1(\mathbf{x}^\star), \qquad \mathbf{b}_2 = \mathbf{C}_2(\mathbf{x}^\star).$$

At $\mathbf{x}^\star$, let us linearize the problem **Phase I** by a perturbation $\boldsymbol{\xi}$. The linearized problem writes

$$\min\ (\mathbf{H}\boldsymbol{\xi} - \mathbf{h})^T\boldsymbol{\Sigma}^{-1}(\mathbf{H}\boldsymbol{\xi} - \mathbf{h}) \qquad \mathbf{s.t.} \quad \mathbf{A}_1\boldsymbol{\xi} = \mathbf{b}_1. \tag{4.32}$$

Since $\mathbf{x}^\star$ is optimal for **Phase I**, the solution of the above linearized problem is $\boldsymbol{\xi}^\star = \mathbf{0}$. Now if we linearize the constraint $\mathbf{C}_2(\mathbf{x}) = \mathbf{0}$ to $\mathbf{A}_2\boldsymbol{\xi} = \mathbf{b}_2$ at $\mathbf{x}^\star$ and add it in (4.32), then by the result in the linear case, i.e. (4.23),

$$\begin{aligned}
\Delta f &= (\mathbf{A}_2\boldsymbol{\xi}^\star - \mathbf{b}_2)^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\boldsymbol{\xi}^\star)\mathbf{A}_2^T]^{-1}(\mathbf{A}_2\boldsymbol{\xi}^\star - \mathbf{b}_2) \\
&\xrightarrow{\boldsymbol{\xi}^\star=\mathbf{0}} \mathbf{b}_2^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\boldsymbol{\xi}^\star)\mathbf{A}_2^T]^{-1}\mathbf{b}_2 \\
&\xrightarrow{\mathbb{C}\mathrm{ov}(\boldsymbol{\xi}^\star)=\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)} \mathbf{C}_2(\mathbf{x}^\star)^T[\mathbf{A}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}_2^T]^{-1}\mathbf{C}_2(\mathbf{x}^\star)_2
\end{aligned}$$

which completes the proof.

The last equality holds because the covariance of $\mathbf{x}^\star$ is defined in the tangent space, i.e.,

$$\mathbb{C}\text{ov}(\boldsymbol{\xi}^\star) = \mathbf{Q} - \mathbf{Q}\mathbf{A}_1^T(\mathbf{A}_1\mathbf{Q}\mathbf{A}_1^T)^{-1}\mathbf{A}_1\mathbf{Q} = \mathbb{C}\text{ov}(\mathbf{x}^\star)$$

with the matrix $\mathbf{Q} = \mathbf{H}^{-1}\boldsymbol{\Sigma}\mathbf{H}^{-T}$.

4.4.3 Accuracy on Nonlinear Extensions

In contrast to the linear case, the formulas (4.29)(4.31) for predicting the change of optimal values in nonlinear cases are not exact anymore.

Regarding the previous problem, the linearization is applied at its optimal solution, i.e., $\mathbf{x}^\star$, thus the cost of the linearized problem is its optimal value $f^\star$. The approximation occurs at the next/second problem while we literally do not solve it at all! To see this, let us denote $f_{1st}^{\star\star}$ as the cost of the linearized problem after optimizing the second problem by only one iteration. Then in essence, the equation (4.29) and (4.31) approximate the actual change of optimal values, i.e., Δf, by $\Delta f \approx f_{1st}^{\star\star} - f^\star$.

Now we reach the status to conclude that: The accuracy of the approximation is determined by the closeness between $f_{1st}^{\star\star}$ and the actual optimal value of the second/next problem $f_{1st}^{\star\star}$. In Chapter 5, we will show experimentally by several optimization instances that the gap turns out to be pretty small. However, we believe that a thorough theoretical analysis on $f_{1st}^{\star\star}$ and $f^{\star\star}$ would definitely lead to some interesting results, and we leave this part as a further research direction.

4.5 Discussions

4.5.1 State Variable with Increasing Dimensions

In previous sections, we prove our results by assuming that the dimension of the state variable $\mathbf{x}$ remains unchanged. However, it is pretty easy to extend the result derived in previous

sections, to the case of $\mathbf{x}$ with increasing dimensions, which is actually a typically scenario in incremental optimization problems.

Without loss of generality, let us assume after adding a new measurement, say $\tilde{\mathbf{z}}$, the state variable $\mathbf{x}$ is augmented to $[\mathbf{x}^T\ \mathbf{v}^T]^T$. **Obviously, by adding $\tilde{\mathbf{z}}$ only, the newly added variable $\mathbf{v}$ is unconstrained thus the resulting change of optimal values is exactly zero.** This is the key insight that we would like to exploit when dealing with measurements that increase the dimension of state variables.

An extension to the previous insight is: In case that there are a set of measurements $\{\tilde{\mathbf{z}}_i\}_{i\in N}$, where each $\tilde{\mathbf{z}}_i$ would introduce a **different** variable to the optimization problem, then the change of optimal values by adding $\{\tilde{\mathbf{z}}_i\}_{i\in N}$ is exactly zero. Nevertheless, we can also check $\{\tilde{\mathbf{z}}_i\}_{i\in N}$ one by one in sequence, and just apply the conclusion on each single measurement.

4.5.2 Exploiting Sparsity

The calculation of the change of optimal values Δf requires to compute a matrix in the form of $[\mathbf{A}\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}^T]^{-1}$, where $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$ constitute a bottleneck of the computation. However, in real applications, $\mathbf{A}$ usually possesses certain sparsity. In this case, we do not need to compute the full covariance matrix $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$, but instead a marginal covariance matrix with respect to the non-zeros blocks in $\mathbf{A}$. For example, let us consider $\mathbf{A}$ in the form of

$$\mathbf{A} = \begin{bmatrix} \cdots & \mathbf{J}_i & \cdots & \mathbf{J}_j & \cdots & \mathbf{J}_k & \cdots \end{bmatrix}.$$

By exploiting the sparsity of $\mathbf{A}$, we expand $\mathbf{A}\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}^T$ as below,

$$
\begin{aligned}
&\mathbf{A}\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}^T \\
&= \begin{bmatrix} \cdots & \mathbf{J}_i & \cdots & \mathbf{J}_j & \cdots & \mathbf{J}_k & \cdots \end{bmatrix}
\begin{bmatrix}
 & \vdots & & \vdots & & \vdots & \\
\cdots & \boldsymbol{\Sigma}_{i,i} & \cdots & \boldsymbol{\Sigma}_{i,j} & \cdots & \boldsymbol{\Sigma}_{i,k} & \cdots \\
 & \vdots & & \vdots & & \vdots & \\
\cdots & \boldsymbol{\Sigma}_{j,i} & \cdots & \boldsymbol{\Sigma}_{j,j} & \cdots & \boldsymbol{\Sigma}_{j,k} & \cdots \\
 & \vdots & & \vdots & & \vdots & \\
\cdots & \boldsymbol{\Sigma}_{k,i} & \cdots & \boldsymbol{\Sigma}_{k,j} & \cdots & \boldsymbol{\Sigma}_{k,k} & \cdots \\
 & \vdots & & \vdots & & \vdots &
\end{bmatrix}
\begin{bmatrix} \vdots \\ \mathbf{J}_i^T \\ \vdots \\ \mathbf{J}_j^T \\ \vdots \\ \mathbf{J}_k^T \\ \vdots \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{J}_i & \mathbf{J}_j & \mathbf{J}_k \end{bmatrix}
\underbrace{\begin{bmatrix}
\boldsymbol{\Sigma}_{i,i} & \boldsymbol{\Sigma}_{i,j} & \boldsymbol{\Sigma}_{i,k} \\
\boldsymbol{\Sigma}_{j,i} & \boldsymbol{\Sigma}_{j,j} & \boldsymbol{\Sigma}_{j,k} \\
\boldsymbol{\Sigma}_{k,i} & \boldsymbol{\Sigma}_{k,j} & \boldsymbol{\Sigma}_{k,k}
\end{bmatrix}}_{\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)_{i,j,k}}
\begin{bmatrix} \mathbf{J}_i^T \\ \mathbf{J}_j^T \\ \mathbf{J}_k^T \end{bmatrix}.
\end{aligned}
$$

Hence, to compute $\mathbf{A}\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{A}^T$, what we need is the marginal covariance matrix $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)_{i,j,k}$. The marginal covariance matrix can be computed much more efficiently than the full covariance matrix $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)$, particularly in an incremental scheme. See [91] for a reference on some state-of-the-art techniques to compute $\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)_{i,j,k}$.

4.5.3 Information Gain and Change of Optimal Values

Consider the weighted linear least squares in Section 4.3.1. Let us recall that the formula for the change of optimal values Δf writes

$$\Delta f = (\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2)^T[\mathbf{H}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{H}_2^T + \boldsymbol{\Sigma}_2]^{-1}(\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2).$$

In the literature of the information gain [90], the matrix in the middle is typically called the Innovation Covariance Matrix, defined as

$$\mathbf{S} = \mathbf{H}_2\mathbb{C}\mathrm{ov}(\mathbf{x}^\star)\mathbf{H}_2^T + \boldsymbol{\Sigma}_2. \tag{4.33}$$

The term $\mathbf{e}_2 \overset{\text{def.}}{=\!=} \mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2$ can be interpreted as the measurement error of the data $< \mathbf{H}_2, \mathbf{h}_2 >$ when evaluated at the solution $\mathbf{x}^\star$. With a slight abuse of notations, we write

$$\Delta f = (\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2)^T\mathbf{S}^{-1}(\mathbf{H}_2\mathbf{x}^\star - \mathbf{h}_2) = \mathbf{e}_2^T\mathbf{S}^{-1}\mathbf{e}_2. \tag{4.34}$$

The Fisher information matrix of a maximum likelihood estimation is presented as the inverse of the covariance matrix [102]. In case of the weighted linear least squares in Section 4.3.1, the Fisher information matrix without the measurement $< \mathbf{H}_2, \mathbf{h}_2 >$ is $\mathbf{\Lambda} = \mathbb{C}\text{ov}(\mathbf{x}^\star)^{-1} = \mathbf{H}_1^T\mathbf{\Sigma}_1^{-1}\mathbf{H}_1$, while the Fisher information matrix after including the measurement $< \mathbf{H}_2, \mathbf{h}_2 >$ is $\mathbf{\Lambda}' = \mathbf{\Lambda} + \mathbf{H}_2^T\mathbf{\Sigma}_2^{-1}\mathbf{H}_2 = \mathbf{H}_1^T\mathbf{\Sigma}_1^{-1}\mathbf{H}_1 + \mathbf{H}_2^T\mathbf{\Sigma}_2^{-1}\mathbf{H}_2$.

Let us denote $\Delta\mathcal{I}$ as the information gain [49, 57, 90] caused by adding the measurement $< \mathbf{H}_2, \mathbf{h}_2 >$ into the estimation problem. Note that the information gain is also termed "mutual information" in some references [49]. It has been shown in [90] that the information gain $\Delta\mathcal{I}$ writes

$$\Delta\mathcal{I} = \frac{1}{2}\ln\frac{\det(\mathbf{\Lambda}')}{\det(\mathbf{\Lambda})} \overset{[90]}{=\!=} \frac{1}{2}\ln\frac{\det(\mathbf{S})}{\det(\mathbf{\Sigma}_2)}, \tag{4.35}$$

where $\mathbf{S}$ is the innovation covariance matrix define in (4.33). The proof of the second equality above is a direct deployment of the matrix determinant lemma. The details of the proof can be found in [90].

Conclusion: A comparison of (4.34) and (4.35) shows that: (a) Both $\Delta\mathcal{I}$ and Δf are related to the innovation covariance matrix $\mathbf{S}$. (b) Δf considers the measurement error $\mathbf{e}_2$ while $\Delta\mathcal{I}$ does not. (c) $\Delta\mathcal{I}$ and Δf have similar computational complexity, because the bottleneck lies in the computation of $\mathbf{S}$.

4.5.4 Why Information Gain Solely is Insufficient?

By (4.35), it is clear to see that the information gain $\Delta\mathcal{I}$ does not contain any message on the measurement error $\mathbf{e}_2$. In specific, by checking (4.33)(4.35), we would find that $\Delta\mathcal{I}$ is completely independent of the residual vector $\mathbf{b}$ in the linear case. In nonlinear cases,

through linearization, the vector $\mathbf{b}$ is usually somehow related to the residual error of the measurement, and moreover the impact of $\mathbf{b}$ can be passed to the estimation problem by its impact on the linearization point. Therefore, $\mathbf{b}$ would pose an impact on $\Delta\mathcal{I}$ for nonlinear cases. However, it would be reasonable to presume that this impact is caused by linearization. If other techniques rather than linearization is used, we believe that the vector $\mathbf{b}$ should be exactly irrelevant to the information gain $\Delta\mathcal{I}$ for nonlinear cases as well.

In some applications, the message contained in the measurement error $\mathbf{e}_2$, or the residual $\mathbf{b}$ is actually quite useful. This is because $\mathbf{e}_2$ provides a measure on how well the new data matches the current estimate, and $\mathbf{b}$ provides a measure on how big the residual error is. Let us term this aspect of the estimation as the "**consistency of the measurement**". As discussed above, the information gain $\Delta\mathcal{I}$ is essentially a poor metric in terms of evaluating the consistency of the measurement. On the contrary, the change of optimal values Δf is much well-suited for tasks of this kind. In practice, we would expect better performance for the information gain based applications by using the change of optimal values Δf as a supplement. Furthermore, in general we suggest using $\Delta\mathcal{I}$ and Δf together whenever is possible, since both requires the calculation of $\mathbf{S}$ and share the majority of the computation.

At last, let us visit an interesting result by Khosoussi et al. [102], to further evidence our claim. In [102], it was shown that for SLAM instances where the rotation is known, (named "compass SLAM" in [102], which is actually a linear measurement model), the determinant of the Fisher information matrix is related to the number of spanning trees (denoted by $\mathtt{t}$) in the graph. For graph optimizations, adding a new measurement implies adding a new edge to the graph. Let the original graph be $\mathcal{G}$, and the resulting augmented graph be $\mathcal{G}'$. We have $\det(\mathbf{\Lambda}) \propto \mathtt{t}(\mathcal{G})$, and $\det(\mathbf{\Lambda}') \propto \mathtt{t}(\mathcal{G}')$. In this case, the information gain in (4.35) writes,

$$\Delta\mathcal{I} = \frac{1}{2}\ln\frac{\det(\mathbf{\Lambda}')}{\det(\mathbf{\Lambda})} \propto \ln\frac{\mathtt{t}(\mathcal{G}')}{\mathtt{t}(\mathcal{G})}. \tag{4.36}$$

In (4.36), the information gain $\Delta\mathcal{I}$ is completely determined by the topology of the graph. Therefore, $\Delta\mathcal{I}$ has nothing to do with how accurate or how wrong the measurement is.

4.5.5 Computational Complexity

In incremental scenarios, it is also possible to compute an approximate solution to the second subproblem, exploiting special structure and sparse linear algebraic techniques [43, 71, 95, 96, 132]. While this idea works and is a general practice, in particular for least squares optimization problems, the main insight of the COOV metric over previous publications is that: 1) It is actually possible to derive an analytical equation (in closed form) which quantifies the COOV directly. 2) The closed form equations of the COOV open up new possibilities of computational procedures, i.e., via the quantities appeared in the COOV metric instead of an explicit solution. 3) The COOV metric is more advantageous in face of many new measurement candidates, which can share the computation of $\mathbf{x}_1^\star$ and $\mathbb{C}\text{ov}(\mathbf{x}_1^\star)$.

For the COOV metric, the computational bottleneck of Δf lies in the calculation of $\mathbb{C}\text{ov}(\mathbf{x}_1^\star)$, which can be mitigated by computing a marginal covariance matrix [16, 91], i.e., only computing the blocks required by $\mathbf{A}_2\mathbb{C}\text{ov}(\mathbf{x}_1^\star)\mathbf{A}_2^T$ explicitly. There are works by calculating the blocks in $\mathbb{C}\text{ov}(\mathbf{x}_1^\star)$ from matrix factorizations based on recursive formulas [91, 94]. It is also possible to incrementally maintain a marginal covariance matrix by updating $\mathbb{C}\text{ov}(\mathbf{x}_1^\star)$ using the matrix inverse lemma [91]. Last but not least, for some applications, the covariance matrix is already computed, thus the COOV metric can be integrated in with no effort, for example, the applications based on the information gain [90].

It is also of great interests to further compare the numerical procedure to calculate an updated solution to the second subproblem, with that to calculate $\mathbf{A}_2\mathbb{C}\text{ov}(\mathbf{x}_1^\star)\mathbf{A}_2^T$. Further research in this direction will help understanding the connection between updating the solution and updating the optimal value.

Chapter Five

Derived Algorithms and Applications

In this chapter, we propose three applications based on the COOV metric to demonstrate the potential of our theory. We have to admit that the applications shown here are just a tip of the iceberg, and more can be expected which is beyond the scope of this thesis. Therefore we leave these possibilities as a future work and open to the community.

5.1 Outlier Detection in PGO

As the first application, we will take the classical vertex-based PGO as an example, and show in detail how to calculate the COOV metric. Several numerical examples on standard PGO benchmarks are provided to validate the accuracy of the COOV metric. Then we show the usefulness of COOV in terms of discerning outliers, with a comparison against M-estimators.

5.1.1 The Metric to Predict the Change of Optimal Values

For PGO, the measurement model is $\mathbf{m}(\mathbf{T}_i, \mathbf{T}_j) = \mathbf{T}_i^{-1}\mathbf{T}_j$. Let the corresponding measurement data be $\tilde{\mathbf{T}}_{i,j}$, with covariance matrix $\tilde{\mathbf{\Sigma}}_{i,j}$ defined in the tangent space. In specific, we consider the measurement error function

$$\boldsymbol{\tau}_{i,j} = \mathbf{m}(\mathbf{T}_i, \mathbf{T}_j) \boxminus \tilde{\mathbf{T}}_{i,j} = \mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1} \cdot \mathbf{T}_i^{-1}\mathbf{T}_j)$$

$$\boldsymbol{\tau}_{i,j} \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{\Sigma}}_{i,j})$$

where $\boldsymbol{\tau}_{i,j}$ is assumed to follow a zero-mean Gaussian with the covariance matrix given by $\tilde{\boldsymbol{\Sigma}}_{i,j}$. At the current step, the estimates of $\mathbf{T}_i$ and $\mathbf{T}_j$ are given as $\mathbf{T}_i^\star$, $\mathbf{T}_j^\star$. Let $\mathbb{C}\mathrm{ov}(\cdot)$ be the covariance matrix of the solution (for all the poses). The covariance matrix for the estimate $\mathbf{T}_i^\star$, $\mathbf{T}_j^\star$ (in the tangent space) is the i-th and j-th block row and block column of $\mathbb{C}\mathrm{ov}(\cdot)$. This matrix is typically termed "marginal covariance matrix" in robotics, which writes,

$$\mathbb{C}\mathrm{ov}_{i,j} = \begin{bmatrix} \boldsymbol{\Sigma}_{i,i} & \boldsymbol{\Sigma}_{i,j} \\ \boldsymbol{\Sigma}_{j,i} & \boldsymbol{\Sigma}_{j,j} \end{bmatrix}.$$

Please refer to [91] for the techniques to calculate the marginal covariance matrix.

For PGO, the measurement can be classified into two categories: odometry and loop-closure. The odometry measurement introduces new poses while having no impacts on previous poses, so the COOV after including an odometry measurement is always zero. For a loop-closure measurement, we can linearize the measurement model, and apply the result in (4.29) to predict the COOV.

The linearization of the SE(3) manifold is achieved by adding a perturbation in the tangent space with the exponential mapping, and then compute the differential with respect to the perturbation. Let us apply perturbations $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$ to the right-hand side of $\mathbf{T}_i$ and $\mathbf{T}_j$ respectively. After linearization, $\boldsymbol{\tau}_{i,j}$ takes the form

$$\boldsymbol{\tau}_{i,j} \approx \mathbf{J}_i \cdot \boldsymbol{\xi}_i + \mathbf{J}_j \cdot \boldsymbol{\xi}_j + \boldsymbol{\tau}_{i,j}^\star$$

with $\boldsymbol{\tau}_{i,j}^\star = \mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1}(\mathbf{T}_i^\star)^{-1}\mathbf{T}_j^\star)$. The Jacobian matrices involved are defined by

$$\mathbf{J}_i = \frac{\partial \mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1} \cdot (\mathbf{T}_i \cdot \mathbf{Exp}(\boldsymbol{\xi}_i))^{-1}\mathbf{T}_j)}{\partial \boldsymbol{\xi}_i^T}\Big|_{\boldsymbol{\xi}_i=\mathbf{0}}$$

$$\mathbf{J}_j = \frac{\partial \mathbf{Log}(\tilde{\mathbf{T}}_{i,j}^{-1} \cdot \mathbf{T}_i^{-1}(\mathbf{T}_j \cdot \mathbf{Exp}(\boldsymbol{\xi}_j)))}{\partial \boldsymbol{\xi}_j^T}\Big|_{\boldsymbol{\xi}_j=\mathbf{0}}.$$

An analytical form of these Jacobian matrices can be computed easily, see [16]. Here we write the result directly as below,

$$\mathbf{J}_i = -\boldsymbol{\mathcal{J}}_l^{-1}(\boldsymbol{\tau}_{i,j}^\star) \cdot \mathbf{Ad}(\tilde{\mathbf{T}}_{i,j}^{-1}), \quad \mathbf{J}_j = \boldsymbol{\mathcal{J}}_r^{-1}(\boldsymbol{\tau}_{i,j}^\star).$$

Since the Jacobian of $\boldsymbol{\tau}_{i,j}$ contains only two non-zeros blocks, $\mathbf{J}_i$ and $\mathbf{J}_j$, we can exploit the sparsity and write the "innovation covariance matrix" $\mathbf{S}$ as

$$\mathbf{S} = \begin{bmatrix} \mathbf{J}_i & \mathbf{J}_j \end{bmatrix} \cdot \mathbb{C}\text{ov}_{i,j} \cdot \begin{bmatrix} \mathbf{J}_i & \mathbf{J}_j \end{bmatrix}^T + \tilde{\boldsymbol{\Sigma}}_{i,j}.$$

Then applying the result in (4.29) for a loop-closure measurement, and combining the trivial case of an odometry measurement, the COOV after including a new measurement $\tilde{\mathbf{T}}_{i,j}$ is

$$\Delta f \begin{cases} = 0 & \text{(odometry)} \\ \approx (\boldsymbol{\tau}^\star_{i,j})^T \cdot \mathbf{S}^{-1} \cdot \boldsymbol{\tau}^\star_{i,j} & \text{(loop closure)} \end{cases}. \tag{5.1}$$

5.1.2 Robustness Towards Outliers

Given the change of optimal values, the outlier detection is rather straightforward by introducing a threshold on the COOV for each new measurement. A new measurement is considered as an inlier if its resulting COOV is lower than the threshold, and an outlier otherwise. The threshold can be chosen manually or based on a probability test. Here we use the χ^2 different test [143]. The χ^2 difference test is a variation of the χ^2 test employed on the difference of χ^2 error and the change of the degree of freedom (DOF), which provides a probabilistic interpretation of COOV in the Gaussian regression. While we will show a simple χ^2 difference test works in practice, it will be interesting to apply sophisticated classification techniques on COOV, which we leave as an open problem to the community.

5.1.3 Implementation and Experimental Setup

Implementation

We implement our approach on the SLAM++ graph optimization library [3, 92], which provides state-of-the-art strategy to compute the marginal covariance with real-time performance, particularly in an incremental setting. We opt to use a batch solver (Lambda solver)

in the SLAM++ setting and set the error tolerance for convergence to 1e-5 which attains a good balance between the accuracy and timing. An incremental solver (like FastL solver in SLAM++) with larger error tolerance can be significantly faster, however with some compromise on the accuracy. All the experiments are carried out on an Intel i5-5300U CPU. This is a quad core CPU with 2.30GHz for each, and there is no independent GPU.

Datasets

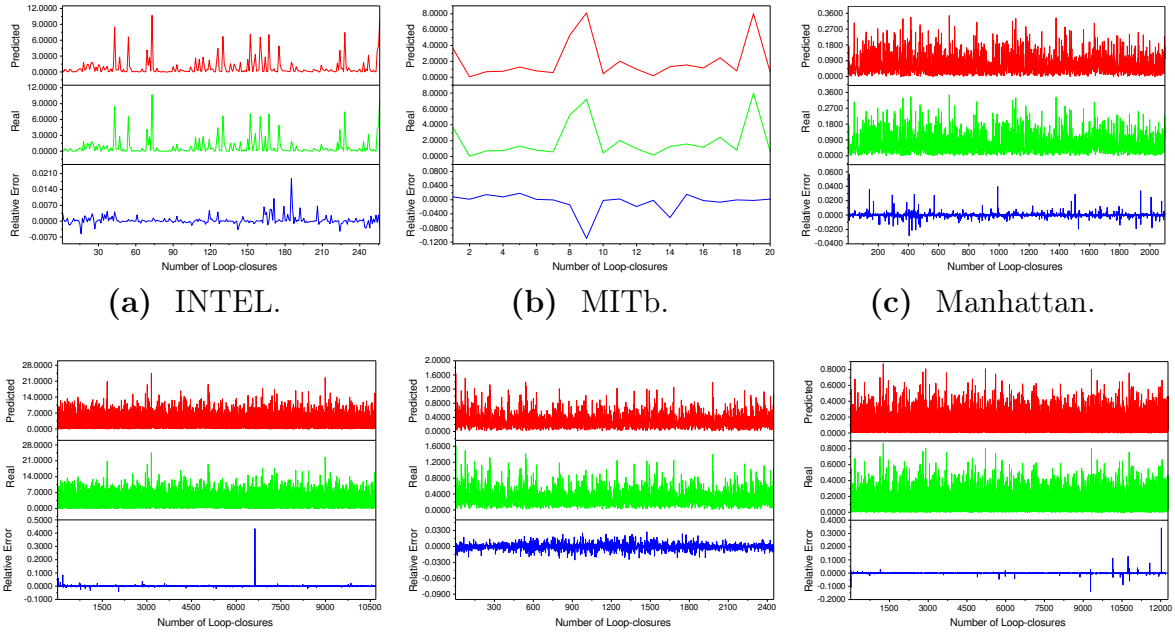
The evaluation is carried out on two real datasets, INTEL [37] and MITb [36], and four simulated datasets, Manhattan [127], City10k [95], Sphere2500 [95] and Torus [95]. Sphere2500 and Torus are 3D datasets while the rest are 2D datasets. The raw data are available at [1], and the details for the data processing are given in [37] and [36] respectively. The brief information of these datasets is reported in Table 5.1.

Table 5.1 Brief information of the datasets used for evaluation.

Dataset	Poses	Loop-Closures	Noise-Level	Timing
INTEL	1228	256	High	5.6s
MITb	808	20	High	1.5s
Manhattan	3500	2099	Medium	150.0s
City10k	10000	10688	Medium	6754.9s
Sphere2500	2500	2450	Medium	848.3s
Torus	10000	12281	Medium	15582.8s

5.1.4 Experiments: Predicting the Change of Optimal Values

For each dataset, we reorder relative pose measurements in an incremental online sequence, to mimic the measurement data sequence obtained in the real online SLAM application. The COOV after including an odometry measurement is trivially zero, so we only evaluate the COOV after including a loop-closure measurement.



(a) INTEL. **(b)** MITb. **(c)** Manhattan.

(d) City10k. **(e)** Sphere2500. **(f)** Torus.

Figure 5.1 The predicted and real COOV after including each loop-closure measurement. The peak in City10k (the point with relative error 43%) is caused by the numerical precision, where the predicted and real COOV are 0.001462 and 0.002097 respectively. However, by using smaller error tolerance for convergence like 1e-6, the values can be reduced to 0.001462 and 0.001410, with relative error around 3.6%.

Accuracy

We feed the data one by one into the SLAM++ solver, and record respectively the predicted and the real COOV after including a loop-closure measurement. We use the relative error

$$\gamma = \frac{\Delta f_{real} - \Delta f_{predicted}}{\Delta f_{predicted}}$$

to quantify the correctness of the proposed metric in (5.1). All these statistics are reported in Fig. 5.1. It can be seen from Fig. 5.1 that most of the time, the relative error is pretty close to zero. The largest relative error of the prediction lies around $5\% \sim 10\%$, which is accurate enough to be used for applications like outlier detection.

Timing

The total timing for each dataset is reported in the last column of Table 5.1. The computation is roughly in real-time for each measurement data. However, as already mentioned, we use a batch solver to benchmark the accuracy of the prediction. For real applications, the timing can be improved by using an incremental solver. According to our tests on SLAM++, FastL solver can be at least 2-3 times faster if using a larger error tolerance for example 0.001 (instead of 1e-5).

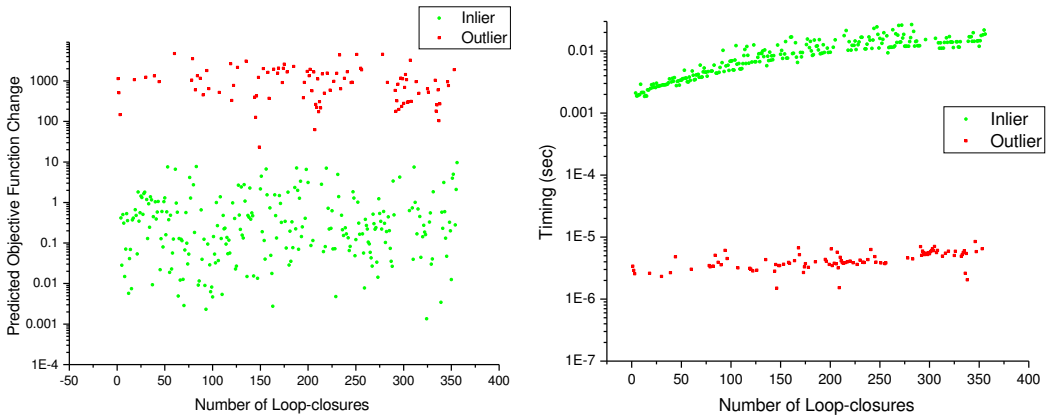
5.1.5 Experiments: Outlier Detection by the Change of Optimal Values

At last, we show how to detect outliers by the COOV. We use the INTEL dataset for the evaluation. The original INTEL dataset is outlier-free, and we take the optimal solution as the ground truth. The outliers are generated by randomly selecting two poses, computing their relative pose measurement with the ground truth, and then assigning the computed relative pose measurement randomly to two other poses. The absolute trajectory error (ATE) is computed as the Euclidean norm of the Cartesian difference between the solution and the ground truth.

Illustration of Outlier Detection

At first, we add 100 outliers to the INTEL dataset, and organize the data sequentially. Then we incrementally solve the dataset containing outliers, using the predicted COOV as a benchmark to discern outliers. The predicted COOV for the inlier and outlier loop-closure measurement is presented in Fig. 5.2a. It can be seen that the predicted COOV for an outlier is usually orders of magnitude larger than that for an inlier. Thus the inlier and outlier measurement can be easily classified by a simple χ^2 difference test [15, 143].

The computational time for each measurement is reported in Fig. 5.2b, where the timing



(a) Change of Optimal Values.

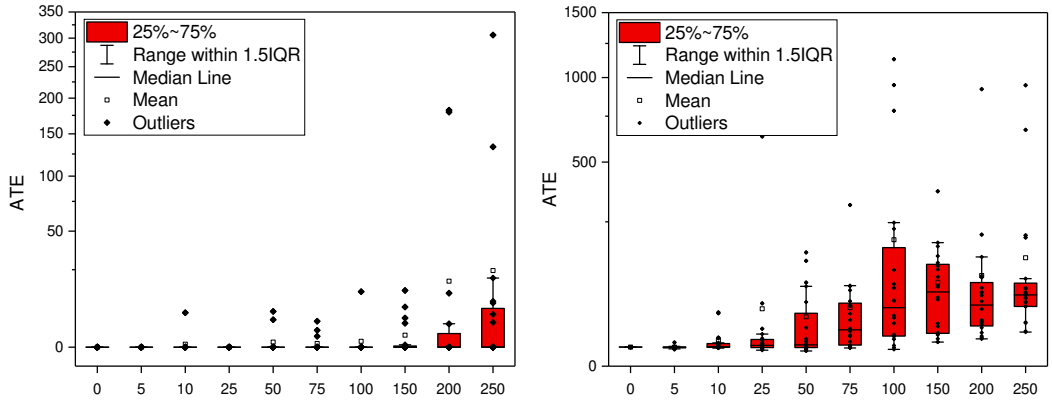
(b) Computational time.

Figure 5.2 An example of the predicted COOV for the inlier and outlier loop-closure measurement and the overall computational time for each measurement on the INTEL dataset.

for outliers is negligible compared with that for inliers. This is because for an outlier measurement, we only need to compute the metric to predict the COOV, without any additional expense (namely solving the problem). This is substantially different from the existing χ^2 test based methods [75, 106], where a problem containing outliers is solved which degenerates the efficiency of the algorithm.

Robustness and Timing on Monte-Carlo Simulation

To evaluate the robustness of our method against outliers, we use a 20 run Monte-Carlo simulation, with respect to different quantity of outliers. We compare our method with Cauchy M-estimator (with tuning parameter set to 1) in an online incremental scheme. We disable the covariance computation for Cauchy (which is not required for M-estimators and saves a large amount of time), and set the error tolerance for convergence to 1e-5 for a fair comparison. The resulting ATE is plotted in Fig. 5.3. It can be seen that our method (shown in Fig. 5.3a) diverges less and produces more accurate results than Cauchy (shown in Fig. 5.3b). The overall computational time of our method and Cauchy for each Monte-Carlo trial is sketched in Fig. 5.4. In general, our method is much faster than Cauchy M-estimator



(a) Ours. (b) Cauchy.

Figure 5.3 The robustness of our method and Cauchy M-estimator on the INTEL dataset, shown as the absolute trajectory error (ATE) with respect to different number of outliers on a 20 run Monte-Carlo simulation.

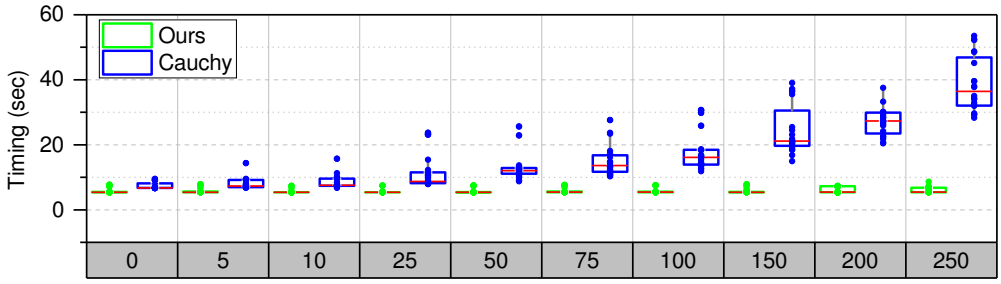


Figure 5.4 The overall computational time of our method and Cauchy M-estimators for each Monte-Carlo run on the INTEL dataset.

in an online incremental scheme, and our method scales well with the growing number of outliers, contrasted to the radical growth in timing of Cauchy.

5.2 Cost of Aligning Two Trajectories

The second application is to forecast the optimal value of an optimization problem by the COOV metric directly. This is particularly useful when the previous subproblem has a trivial solution. To show the point, this section will formulate a nonlinear least distance optimization problem to align two trajectories, and show experimentally the accuracy of the COOV metric. The Matlab code also shows that computing the optimal value by the COOV

metric is also magnitudes faster than solving problems directly. The author believes that the timing difference can be larger if using a C++ implementation.

5.2.1 Problem Statement and Theoretical Results

A trajectory is, in its explicit form, a collection of poses. Let $\mathcal{T}$ be a trajectory, then it can be described by a union of poses, $\mathcal{T} = \cup \mathbf{T}_i$, $\mathbf{T}_i \in \mathrm{SE}(d)$, $d = 2, 3$. However, in many applications, if one pose is moved, we want the impact to be passed to other poses as well. Therefore, it is more desirable to exploit an alternative parameterization based on relative transformations (termed relative poses in robotics). Without loss of generality, let us define a trajectory as a head pose plus a chain of relative poses following the head pose. Note that in this case, the relative poses in the trajectory are subject to a certain order, which can be identified by the subscripts of relative poses. Let $\mathcal{T}_A$ and $\mathcal{T}_B$ be two trajectories given by

$$\mathcal{T}_A \stackrel{\text{def}}{=} {}^A\mathbf{T}_1, {}^A\mathbf{T}_{1,2}, {}^A\mathbf{T}_{2,3}, \dots {}^A\mathbf{T}_{N_A,N_A+1}$$

$$\mathcal{T}_B \stackrel{\text{def}}{=} {}^B\mathbf{T}_1, {}^B\mathbf{T}_{1,2}, {}^B\mathbf{T}_{2,3}, \dots {}^B\mathbf{T}_{N_B,N_B+1}$$

where ${}^A\mathbf{T}_1$ is the head pose of trajectory $\mathcal{T}_A$ and ${}^A\mathbf{T}_{i,i+1}$ $(i = 1, 2, \cdots N_A)$ is a set of relative poses following the head pose ${}^A\mathbf{T}_1$. The same rule applies to the trajectory $\mathcal{T}_B$.

In this parameterization, we obtain any arbitrary pose by chaining out the relative poses. For example, in the trajectory $\mathcal{T}_A$, an arbitrary pose ${}^A\mathbf{T}_k$ can be written as

$${}^A\mathbf{T}_k = {}^A\mathbf{T}_1 {}^A\mathbf{T}_{1,2} \cdots {}^A\mathbf{T}_{k-1,k}.$$

Now at some point, we know that the l-th pose ${}^A\mathbf{T}_l$ in the trajectory $\mathcal{T}_A$ and the r-th pose ${}^B\mathbf{T}_r$ in the trajectory $\mathcal{T}_B$ are the same pose, which results to a hard constraint ${}^A\mathbf{T}_l = {}^B\mathbf{T}_r$

and an optimization problem in the form of

$$\begin{aligned}
\min\ & \underbrace{\|{}^{A}\mathbf{T}_1 \boxminus {}^{A}\tilde{\mathbf{T}}_1\|^2_{A\Sigma_1} + \sum_{i=1}^{N_A} \|{}^{A}\mathbf{T}_{i,i+1} \boxminus {}^{A}\tilde{\mathbf{T}}_{i,i+1}\|^2_{A\Sigma_{i,i+1}}}_{\text{Cost of trajectory } \mathcal{T}_A} \\
& + \underbrace{\|{}^{B}\mathbf{T}_1 \boxminus {}^{B}\tilde{\mathbf{T}}_1\|^2_{B\Sigma_1} + \sum_{j=1}^{N_B} \|{}^{B}\mathbf{T}_{j,j+1} \boxminus {}^{B}\tilde{\mathbf{T}}_{j,j+1}\|^2_{B\Sigma_{j,j+1}}}_{\text{Cost of trajectory } \mathcal{T}_B}. \qquad (5.2) \\
\mathbf{s.t.}\quad & {}^{A}\mathbf{T}_l = {}^{B}\mathbf{T}_r \\
& \text{with } {}^{A}\mathbf{T}_l = {}^{A}\mathbf{T}_1 {}^{A}\mathbf{T}_{1,2} \cdots {}^{A}\mathbf{T}_{l-1,l} \\
& \quad {}^{B}\mathbf{T}_r = {}^{B}\mathbf{T}_1 {}^{B}\mathbf{T}_{1,2} \cdots {}^{B}\mathbf{T}_{r-1,r}
\end{aligned}$$

where we use the notations ${}^{A}\tilde{\mathbf{T}}_1$, ${}^{B}\tilde{\mathbf{T}}_1$, ${}^{A}\tilde{\mathbf{T}}_{i,i+1}$, ${}^{B}\tilde{\mathbf{T}}_{i,i+1}$ to describe the initial configuration of $\mathcal{T}_A$ and $\mathcal{T}_B$ without the hard constraint ${}^{A}\mathbf{T}_l = {}^{B}\mathbf{T}_r$, while ${}^{A}\mathbf{T}_1$, ${}^{B}\mathbf{T}_1$, ${}^{A}\mathbf{T}_{i,i+1}$, ${}^{B}\mathbf{T}_{i,i+1}$ are state variables to be estimated by imposing the hard constraint ${}^{A}\mathbf{T}_l = {}^{B}\mathbf{T}_r$.

Remark 9. The problem described in (5.2) is essentially an extension to the problem "trajectory bending" studied by Dubbelman et al. [59, 60]. Please refer to [59, 60] for a reference of its application backgrounds. Here in (5.2), we define the "bending" as a more general constraint by aligning any arbitrary two poses together; while the original paper was to "bend" a pose in a trajectory to a specific given value. However mathematically, I think this difference is marginal and one should extend to another. Nevertheless, it is worth noticing how seemingly different problems are essentially a description of kind of the same thing.

Problem: Due to the process noise, there might exist many potential candidate pairs for ${}^{A}\mathbf{T}_l$ and ${}^{B}\mathbf{T}_r$. The question is: What is the cost for each specific pair? A brutal force answer is to solve the optimization problem in (5.2) for each pair. However, alternatively we can use the COOV metric to predict the change of optimal values directly.

In (5.2), let us choose the difference/subtraction operation $\boxminus$ as the logarithm mapping of SE(3) in the vector space of its Lie algebra $\mathfrak{se}(3)$. To apply the result in (4.31), we further

define the addition operation $\boxplus$ by the exponential mapping of SE(3). Formally these two binary operators are defined as

$$\boxminus: \quad \mathbf{T}_1 \boxminus \mathbf{T}_2 = \mathbf{Log}(\mathbf{T}_1^{-1}\mathbf{T}2), \quad \mathbf{T}_1, \mathbf{T}_2 \in \mathrm{SE}(3)$$

$$\boxplus: \quad \mathbf{T} \boxplus \boldsymbol{\xi} = \mathbf{T} \cdot \mathbf{Exp}(\boldsymbol{\xi}), \quad \mathbf{T} \in \mathrm{SE}(3), \boldsymbol{\xi} \in \mathbb{R}^6.$$

The constraint ${}^A\mathbf{T}_l = {}^B\mathbf{T}_r$ can be written in the standard form

$${}^A\mathbf{T}_l = {}^B\mathbf{T}_r \Longleftrightarrow \mathbf{Log}({}^A\mathbf{T}_l {}^B\mathbf{T}_r^{-1}) = \mathbf{0}.$$

Let us denote $\mathbf{C}_2(\mathbf{x}) = \mathbf{Log}({}^A\mathbf{T}_l {}^B\mathbf{T}_r^{-1})$. Now we are at the stage to apply the result in (4.31) directly.

Previous solution and its covariance: Without the constraint ${}^A\mathbf{T}_l = {}^B\mathbf{T}_r$, the optimal trajectory for $\mathcal{T}_A$ and $\mathcal{T}_B$ are simply their original forms

$$\mathcal{T}_A: \quad {}^A\mathbf{T}_1^\star = {}^A\tilde{\mathbf{T}}_1, \quad {}^A\mathbf{T}_{i,i+1}^\star = {}^A\tilde{\mathbf{T}}_{i,i+1} \quad (i = 1, 2, \cdots N_A)$$

$$\mathcal{T}_B: \quad {}^B\mathbf{T}_1^\star = {}^B\tilde{\mathbf{T}}_1, \quad {}^B\mathbf{T}_{j,j+1}^\star = {}^B\tilde{\mathbf{T}}_{j,j+1} \quad (j = 1, 2, \cdots N_B).$$

Hence $f^\star = 0$. Let us collect the above result in the variable $\mathbf{x}_1^\star$. To calculate its covariance matrix $\mathbb{C}\mathrm{ov}(\mathbf{x}_1^\star)$, we linearize the cost function at $\mathbf{x}_1^\star$. By some trivial linear algebra on Lie group, we get the covariance matrix

$$\mathbb{C}\mathrm{ov}(\mathbf{x}_1^\star) = \mathbf{H}^{-1}\boldsymbol{\Sigma}\mathbf{H}^{-T} \quad \text{(by (4.24))}$$

with $\mathbf{H} = \mathbf{I}$, and $\boldsymbol{\Sigma}$ constructed diagonally with corresponding ${}^A\Sigma_i$, ${}^A\Sigma_{i,i+1}$, ${}^B\Sigma_i$, ${}^B\Sigma_{j,j+1}$.

New Constraint: For the constraint, let us index the i-th component of a trajectory, for example the trajectory $\mathcal{T}_A$ as $\mathcal{T}_A[i]$, which basically means $\mathcal{T}_A[1] = {}^A\mathbf{T}_1$, $\mathcal{T}_A[2] = {}^A\mathbf{T}_{1,2}$, $\mathcal{T}_A[3] = {}^A\mathbf{T}_{2,3}$ and so forth. Let us further denote

$$\mathbf{C}_2(\mathbf{x}_1^\star) = \mathbf{Log}({}^A\tilde{\mathbf{T}}_l {}^B\tilde{\mathbf{T}}_r^{-1}) = \boldsymbol{\eta}.$$

The linearization of $\mathbf{C}_2(\mathbf{x}_1)$ at $\mathbf{x}_1^\star$ consists of trivial linear algebras based on the BCH formula and the adjoint operation of SE(3) [19, 45]. We provide the analytical Jacobian $\mathbf{A}_2 =$

$-\frac{\partial \mathbf{C}_2(\mathbf{x}^\star \boxplus \boldsymbol{\xi})}{\partial \boldsymbol{\xi}^T}|_{\boldsymbol{\xi}=\mathbf{0}}$ here by writing down its non-zero blocks, and ignore the calculation details for simplicity:

$$\mathbf{A}_2(\mathcal{T}_A[i]) = -\mathbf{J}_l(\boldsymbol{\eta})^{-1}\mathbf{Ad}({}^A\mathbf{T}_i^\star),\ (i = 1, 2, \cdots l)$$

$$\mathbf{A}_2(\mathcal{T}_B[j]) = \mathbf{J}_l(\boldsymbol{\eta})^{-1}\mathbf{Ad}({}^A\tilde{\mathbf{T}}_l{}^B\tilde{\mathbf{T}}_r^{-1B}\mathbf{T}_{r-j+1}^\star)),\ (j = 1, 2, \cdots r)$$

where $\mathbf{A}_2(\mathcal{T}_A[i])$ and $\mathbf{A}_2(\mathcal{T}_B[j])$ are the blocks corresponding to the i-th component of $\mathcal{T}_A$ and the j-th component of $\mathcal{T}_B$ respectively. $\mathbf{J}_l(\cdot)$, and $\mathbf{Ad}(\cdot)$ are the left-hand Jacobian and the adjoint operation of SE(3) respectively.

Apply Eq. (4.31): Finally, the change of optimal values Δf can be calculated by Eq. (4.31), and the optimal value of the problem (5.2) can be approximated by $f^{\star\star} \approx f^\star + \Delta f = \Delta f$ (since $f^\star = 0$ in this case). Note that for this specific example, we do not need to solve any optimization problem at all. The only requirement is to linearize the constraints, and then apply (4.31)! Thus calculating $f^{\star\star}$ by the COOV metric can be magnitudes cheaper than solving the problem directly.

5.2.2 Experimental Results

In this section, we provide numerical experiments for the trajectory alignment problem defined in (5.2). We use Matlab to simulate two random robot trajectories, by firstly rotating the robot by a random angle, and then moving the robot $1m$ forward. These two trajectories are intersected in the middle in case they are completely apart from one other. Afterwards, we add Gaussian additive noise with standard deviation $0.1m$ on the translational part, and $0.01rads$ on the rotational part. We use an Intel i5-5300U CPU @ 2.30GHz × 4, with Ubuntu 16.04 LTS distribution to run all the experiments.

An Intuitive Example: Trajectory Change, Predicted Cost and Real Cost

First, let us consider a simple example where each trajectory contains 20 poses. We choose several pose combinations to demonstrate the idea as shown in Fig. 5.5. We visualize the

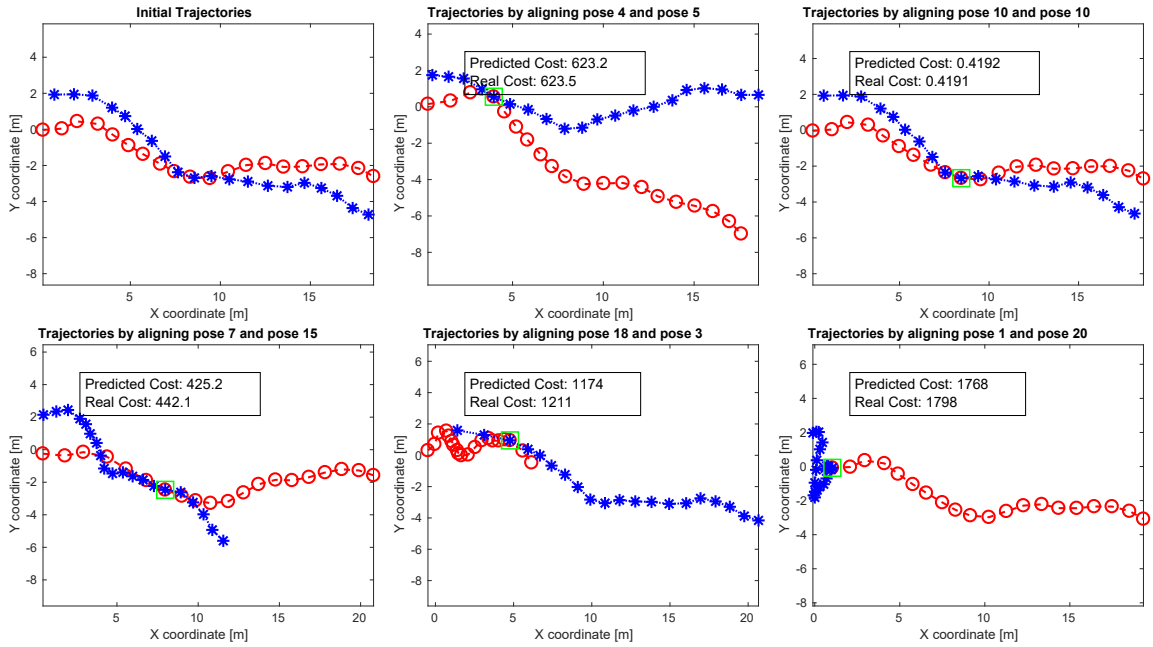


Figure 5.5 Examples of aligning two trajectories, plotted in red and blue respectively. For each aligned case, we show the resulting trajectory and the cost it takes. Both the predicted cost (by the OFC metric) and the real cost (by fully optimizing the problem) are reported for comparison.

change of the trajectory with respect to the cost predicted by the metric, as well as the real cost obtained by completely solving the problem (5.2). From Fig. 5.5, we would say that the metric is quite robust in general; even for the cases of misalignments, like pose 1 and 20, the resulting trajectories are completely twisted, while the metric is still pretty accurate. Note that the initial noise free trajectories are perfectly aligned at the 10-th poses, where the problem attains the lowest cost (with noise) and the metric is particularly precise.

Then we calculate the cost for all possible pose combinations, which is 20×20, and report the corresponding relative error in Fig 5.6. It can be envisioned that there exists many unrealistic alignments within these 20×20 combinations, however the relative error is essentially very low, with a maximum around 10%, which is sufficient for any further decision makings. At last, it takes only 3 seconds to compute the metric for all the 20×20 cases, while the corresponding time consumed by solving the problem is 26 seconds.

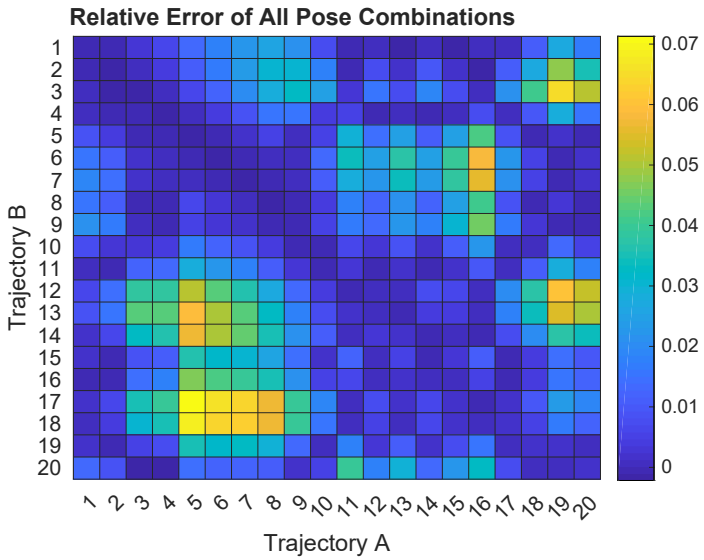


Figure 5.6 The relative error of the predicted cost in each pose combination.

Computational Time and Relative Error with Different Trajectory Lengths

To quantify the effect of trajectory lengths on the accuracy of the metric, we simulate several trajectories with 20, 50, 100 and 200 poses respectively. As before, we compute the alignment for all possible pose combinations for each case. The relative error between the predicted cost (by the metric) and the real cost (by solving the problem) is reported in Fig. 5.7. It can be seen from Fig. 5.7 that for most of the cases, the relative error is still reasonable. However as the length of the trajectory grows, the "unrealistic alignments" (i.e. misalignments) can lead to a substantially amount of outliers in the boxplot. An explanation to this phenomenon is that because the metric is essentially built on the result of linear cases, if the alignment is too far, the linearization point changes dramatically and the linear result is less accurate. However, in general, the predicted cost still provide a reasonable approximation to the real cost.

The computational time is reported in Table 5.2. For each case, we show the overall computational time for all possible pose combinations, as well as the averaged time for obtaining a cost for a single pose combination. Overall, calculating the cost by the metric is at least one magnitude faster than by solving the problem directly. This timing saved here

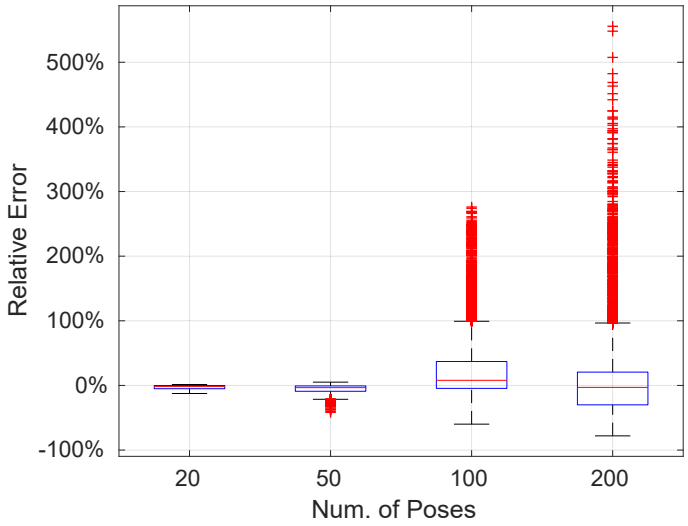


Figure 5.7 The relative error of the metric with respect to different lengths of trajectories. Note here we compute all possible pose combinations, for example the number of possible alignments for 200 poses are 40000. This may of course include some ridiculous combinations, which attributes to the outliers (marked in red) in the plot.

can be extremely beneficially for large problem instances.

5.3 Robust Incremental SLAM in the Cycle Space

At last, we integrate the two cornerstones of this thesis into a robust incremental SLAM framework in the Cycle Space. In specific, we show how to design a robust incremental SLAM framework that is resilient to both local minima and outliers, by using merely the COOV metric. We will formulate SLAM in the cycle space, which extends well to the point feature based SLAM. An alternative cycle basis (besides MCB) is also described in this section. In general, this cycle basis does not guarantee sparsity but is rather cheap to compute. As usual, numerical experiments are provided to backup our claim.

Table 5.2 The Computational Time (Unit: Seconds)

Num. of poses	20	50	100	200
By metric overall	2.49116	36.7919	294.033	2457.14
By solving overall	22.5889	460.228	4799.49	42428.2
By metric avg.	0.00623	0.0147	0.0294	0.0614
By solving avg.	0.0565	0.184	0.480	1.06

5.3.1 Choices of Cycle Basis in an Incremental Scheme

As described in Chapter 3.1, it would always be desirable to choose the cycle basis to be a MCB. However, this choice is expensive, in particular in an incremental scheme when each step we compute a new cycle basis. A compromise is to relax the MCB in the following way:

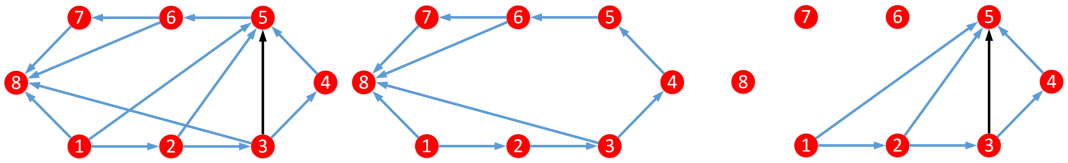
Proposition 3 (Approximate MCB in Incremental Scheme)**.** Let the cycle basis in the previous subproblem $\mathfrak{P}_k$ be denoted as $\mathfrak{B}_k$. In the subproblem $\mathfrak{P}_{k+1}$, if a new edge, say $e_{x,y}$ is added. Then the new cycle basis $\mathfrak{B}_{k+1}$ can be constructed as $\mathfrak{B}_{k+1} = \mathfrak{B}_k \cup \mathcal{C}_{xy}$, with $\mathcal{C}_{xy} = \mathrm{SP}(x, y) + e_{x,y}$. Here $\mathrm{SP}(x, y)$ is the shortest path between the vertex x and y.

The proof is straightforward, since $\mathcal{C}(x, y)$ contains a new edge $e_{x,y}$ which means $\mathcal{C}(x, y)$ is independent of $\mathfrak{B}_k$, while the shortest path $\mathrm{SP}(x, y)$ locally minimizes the length of $\mathcal{C}(x, y)$. However, the idea in Proposition 3 in no way will guarantee the cycle basis to be a MCB, rather a good approximation. The conclusion in Proposition 3 can be used to construct an approximate MCB (i.e., inexact but close to minimal) incrementally.

Another way which is quite cheap, is to use the fundamental cycle basis (FCB), with some tailored designs to minimize the sparsity.

A Variation of FCB with Better Sparsity

Assumption 1. Consider a directed graph with orientations. Let us assume all the edges in the graph are arranged from the node i to j, with $i < j$; otherwise invert it.



(a) An exemplary graph. **(b)** Cycles with end-vertex 8. **(c)** Cycles with end-vertex 5.

Figure 5.8 Diagram of loop closure cycles.

Assumption 2. Let us choose the spanning tree in FCB as the odometry, which is naturally generated in SLAM/PGO.

The Assumption 1 and Assumption 2 identify a set of cycles by the "end-vertices". Taking the toy graph in Fig. 5.8a as an example. We can identify two classes of cycles by the end-vertex 8 (shown in Fig. 5.8b) and the end-vertex 5 (shown in Fig. 5.8c), respectively. The cycles in Fig. 5.8b are independent from those in Fig. 5.8c, because the loop-closure edges ($e_{1,8}$, $e_{3,8}$, $e_{6,8}$) in Fig. 5.8b does not occur in Fig. 5.8c, and vice versa. It can be easily shown that the cycle basis in Fig. 5.8b and the cycle basis in Fig. 5.8c constitute a cycle basis for the graph in Fig. 5.8a. Identifying a MCB for the graph in Fig. 5.8b and Fig. 5.8c are rather easy [15]. Therefore a cycle basis of Fig. 5.8a, can be computed by putting the together the MCB in Fig. 5.8b and the MCB Fig. 5.8c.

The cycle obtained in this way takes the form

$$\mathcal{C} = \{e_{i,i+1}, e_{i+1,i+2}, \cdots, e_{i+m-1,i+m}, e_{i,k}, e_{i+m,k}\} \tag{5.3}$$

with $e_{i,i+1}, e_{i+1,i+2}, \cdots, e_{i+m-1,i+m}$ being a sequence of odometry edges, and $e_{i,k}, e_{i+m,k}$ (at least one of them) being loop-closure edges.

5.3.2 Choices of Pose Parameterization

Vertex-based PGO

The point-feature based SLAM is essentially a special case of pose-graph SLAM, by ignoring the relative measurement on the rotational part [14, 15]. Therefore to incorporate the case

of point feature-based SLAM, we describe a pose to be in $\mathbb{R}^3 \times \mathrm{SO}(3)$. This parameterization is essentially the same as that based on SE(3), except the tangent space of SO(3) and $\mathbb{R}^3$ are decoupled. In specific, we let

$$\mathbf{T}_i = \{\mathbf{t}_i,\ \mathbf{R}_i\} \in \mathbb{R}^3 \times \mathrm{SO}(3), \quad i \in \mathcal{V}.$$

The relative poses from $\mathbf{T}_i$ to $\mathbf{T}_j$ encoded with edge $e_{i,j}$ is described by $\mathbf{T}_{i,j} = \{\mathbf{t}_j^i,\ \mathbf{R}_j^i\}$, where $\mathbf{t}_j^i = \mathbf{R}_i{}^\intercal(\mathbf{t}_j - \mathbf{t}_i)$, $\mathbf{R}_j^i = \mathbf{R}_i{}^\intercal \mathbf{R}_j$. Denote the noisy measurement of the relative poses by $\tilde{\mathbf{T}}_{i,j} = \{\tilde{\mathbf{t}}_j^i,\ \tilde{\mathbf{R}}_j^i\}$, and let us define the measurement error as

$$\mathbf{e}_{i,j} = \begin{bmatrix} \mathbf{t}_j^i - \tilde{\mathbf{t}}_j^i \\ \mathbf{Log}(\tilde{\mathbf{R}}_j^{i\intercal} \cdot \mathbf{R}_j^i) \end{bmatrix}. \tag{5.4}$$

In line with the convention, we assume the the zero-mean Gaussian noise, $\mathbf{e}_{i,j} \sim \mathcal{N}(\mathbf{0}, \Sigma_{i,j})$ is added in the tangent space of $\mathbf{e}_{i,j}$. The vertex based formulation of PGO writes

$$\min \sum_{i,j} \|\mathbf{e}_{i,j}\|^2_{\Sigma_{i,j}} = \sum_{i,j} \mathbf{e}_{i,j}{}^\intercal \cdot \Sigma_{i,j}^{-1} \cdot \mathbf{e}_{i,j}. \tag{5.5}$$

Cycle based PGO

The relative error in (5.4) is a difference between $\mathbf{T}_{i,j}$ and $\tilde{\mathbf{T}}_{i,j}$. In contrast with the difference defined in SE(3), i.e., $\tilde{\mathbf{T}}_{i,j}^{-1}\mathbf{T}_{i,j}$, the translational part and rotational part are decoupled in (5.4). In this case, if we want to extend to relative point feature observations, we can simply ignore the rotational part, and define the measurement error to be $\mathbf{e}_{i,j} = \mathbf{t}_j^i - \tilde{\mathbf{t}}_j^i$.

On the other hand, the cost function defined in (5.5) can be interpreted as a Mahalanobis distance between $\mathbf{T}_{i,j}$ and $\tilde{\mathbf{T}}_{i,j}$. Let us denote this distance as

$$d(\mathbf{x}, \eta)|_\Sigma = \sum_{i,j} d(\mathbf{T}_{i,j}, \tilde{\mathbf{T}}_{i,j})|_{\Sigma_{i,j}} = \sum_{i,j} \mathbf{e}_{i,j}{}^\intercal \cdot \Sigma_{i,j}^{-1} \cdot \mathbf{e}_{i,j}.$$

where $\mathbf{x}$, η, and Σ collect all $\mathbf{T}_{i,j}$, $\tilde{\mathbf{T}}_{i,j}$, and $\Sigma_{i,j}$, respectively.

The loop-closure cycle with respect to (5.3) writes

$$\mathbf{T}_{i,i+1} \oplus \mathbf{T}_{i+1,i+2} \oplus \cdots \oplus \mathbf{T}_{i+m-1,i+m} \oplus \mathbf{T}_{i+m,k} = \mathbf{T}_{i,k} \tag{5.6}$$

whose expansion (5.6) boils down to two equations

$$[\prod_{l=0}^{m-1}\mathbf{R}_{i+l+1}^{i+l}]\mathbf{t}_k^{i+m}+\sum_{t=0}^{m-1}[\prod_{l=0}^{t-1}\mathbf{R}_{i+l+1}^{i+l}]\mathbf{t}_{i+t+1}^{i+t}-\mathbf{t}_k^i=\mathbf{0} \tag{5.7}$$

$$\mathbf{Log}\{[\mathbf{R}_k^i]^\intercal\cdot[\prod_{l=0}^{m-1}\mathbf{R}_{i+l+1}^{i+l}]\mathbf{R}_k^{i+m}\}=\mathbf{0} \tag{5.8}$$

Let us term (5.7) as the translational constraint, and (5.8) as the rotational constraint. It can be easily verified that, for the cycle based formulation, the point-feature based SLAM contains only the translational constraint [14], while a pose-graph SLAM contains both the translational and rotational constraints [15].

Since the point feature based SLAM can be considered as a specialization of PGO. We discuss PGO only in what follows. Let $\mathbf{C}_i(\mathbf{x})=\mathbf{0}$ be the i-th cycle constraint in the cycle based formulation. Then the cycle based PGO formulation writes

$$\mathfrak{P}:\quad\begin{aligned}&\min\quad d(\mathbf{x},\eta)|_\Sigma\\ &s.t.\quad \mathbf{C}_i(\mathbf{x})=\mathbf{0}\quad(i\in D)\end{aligned} \tag{5.9}$$

with the set D containing all the indices of cycle constraints.

5.3.3 Robust Incremental SLAM Based on Change of Optimal Values

Now we are at the point to discuss the robustness of an incremental SLAM in the cycle based formulation. The robustness to outliers has been investigated in Section 5.1. Therefore in the following we mainly focus on how to increase the robustness against local minima.

Incremental SLAM in Cycle Based Formulation

The key of an incremental SLAM algorithm is to design a sequence of subproblems $\mathfrak{P}_k$ $(k=0,1,2\ldots n)$

$$\mathfrak{P}_k:\quad\begin{aligned}&\min\quad d(\mathbf{x},\eta)|_\Sigma\\ &s.t.\quad \mathbf{C}_i(\mathbf{x})=\mathbf{0}\quad(i\in D_k)\end{aligned} \tag{5.10}$$

which only contains a portion of the constraints from the original problem $\mathfrak{P}$. The sets of indices satisfy

$$D_0 \subset D_1 \subset D_2 \cdots D_{n-1} \subset D_n \subseteq D$$

where $D_0 = \emptyset$, indicating none of the constraints is included at subproblem $\mathfrak{P}_0$.

When solving the subproblems sequentially with SQP,

$$\mathbf{x}_0 \xrightarrow[SQP]{\mathfrak{P}_1} \mathbf{x}_1 \xrightarrow[SQP]{\mathfrak{P}_2} \mathbf{x}_2 \cdots \mathbf{x}_{n-1} \xrightarrow[SQP]{\mathfrak{P}_n} \mathbf{x}_n$$

we get a solution sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_n$. If there are no outliers in the dataset, then $D_n = D$. If there are outliers in the dataset, then D_n is designed to be the maximal constraint set which contains no outliers.

The most critical part of the incremental SLAM is how to design the constraint set D_k for each subproblem $\mathfrak{P}_k$, which can guarantee the global convergence between subproblems and ensure outlier free. In this sense, a robust incremental SLAM can be regarded as a constraint selection problem.

Robustness to Local Minima: Least Change of Optimal Values

To avoid local minima, the solution sequence $\{\mathbf{x}_k\}$ should be designed in a way that $\mathbf{x}_{k-1}$, i.e., the solution of subproblem $\mathfrak{P}_{k-1}$ which also serves as the initial value of subproblem $\mathfrak{P}_k$, lies within the basin of convergence of subproblem $\mathfrak{P}_k$. To achieve this, one possible way is to ensure that $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ are as close as possible to each other as discussed by Hu et al. [86]. This heuristics can be described as a least solution change principle.

However, the solution change (i.e., the closeness of solutions) is not available unless we solve the problem $\mathfrak{P}_k$. Instead we can use the COOV

$$\Delta f_k^{k-1} = f_k - f_{k-1} = d(\mathbf{x}_k, \eta)|_\Sigma - d(\mathbf{x}_{k-1}, \eta)|_\Sigma \tag{5.11}$$

as an approximated measure on the solution change. As discussed in Chapter 4, Δf_k^{k-1} can be predicted by Eq. (4.31), thus served as a metric for constraint selection.

In a word, we can enhance the robustness against local minima between $\mathfrak{P}_{k-1}$ and $\mathfrak{P}_k$, by ensuring the least COOV, i.e., Δf_k^{k-1}.

Robust Incremental SLAM Framework

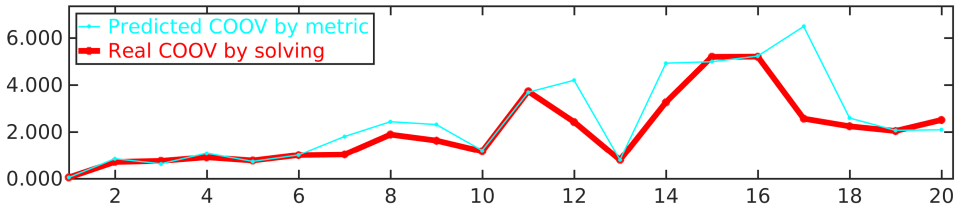
Analogous to Section 5.1, the robustness against outliers can be attained by applying a χ^2 difference test on the COOV. This motivates us an overall robust design for an incremental SLAM, by adding each time a constraint whose induced COOV is minimal and passes the χ^2 different test. In specific, during the constraint selection process, we calculate the COOV for all the (unconsidered) constraints by Eq. (4.31), and select the one that leads to the minimal COOV. A χ^2 difference test is followed to check the validity of the chosen COOV to reject potential outliers. We continue this process until the next best constraint is an outlier, or all constraints are added. In line with Chapter 3, we solve the subproblem in (5.10) by the sequential quadratic programming (SQP). As a result, we will use the term "iSQP" to represent the robust incremental SLAM framework described in this section.

5.3.4 Experimental Results

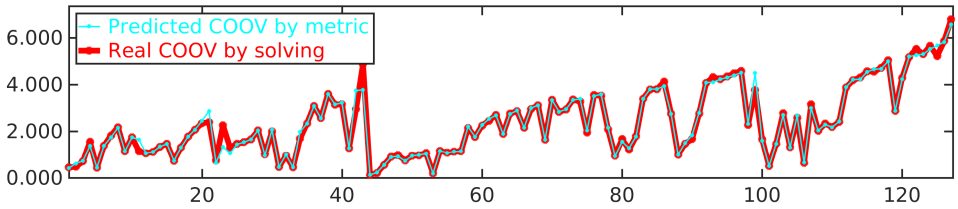
In this section, we provide experimental evaluations on the accuracy of the COOV metric and the overall robustness of iSQP. As usual, we will use M-estimators as the benchmark method, and compare with two of them, i.e., Cauchy and dynamic covariance scaling (DCS) [5]. It has been show M-estimators can also increase the chance of converging to the global minimum [6, 86]. Therefore, we use M-estimators to benchmark the robustness against both local minima and outliers for the proposed iSQP framework.

Accuracy of the COOV Metric

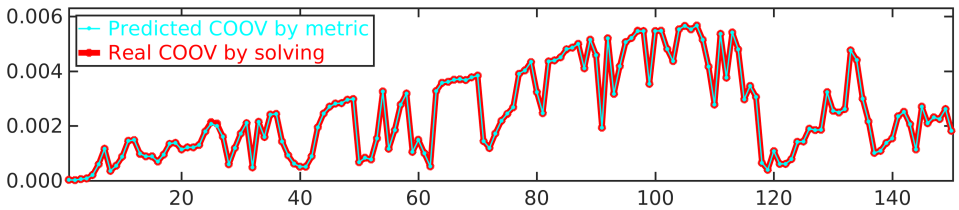
As usual, let us show the accuracy of the COOV metric in terms of forecasting the change of optimal values between subproblems defined in (5.10). To that end, we solve 4 outlier free



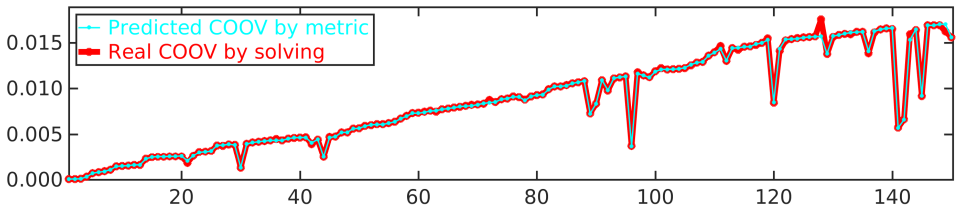
(a) MITb (Total 20 independent constraints).



(b) CSAIL_Pn (Total 127 independent constraints).



(c) Intel (Partial: first 150 independent constraints).



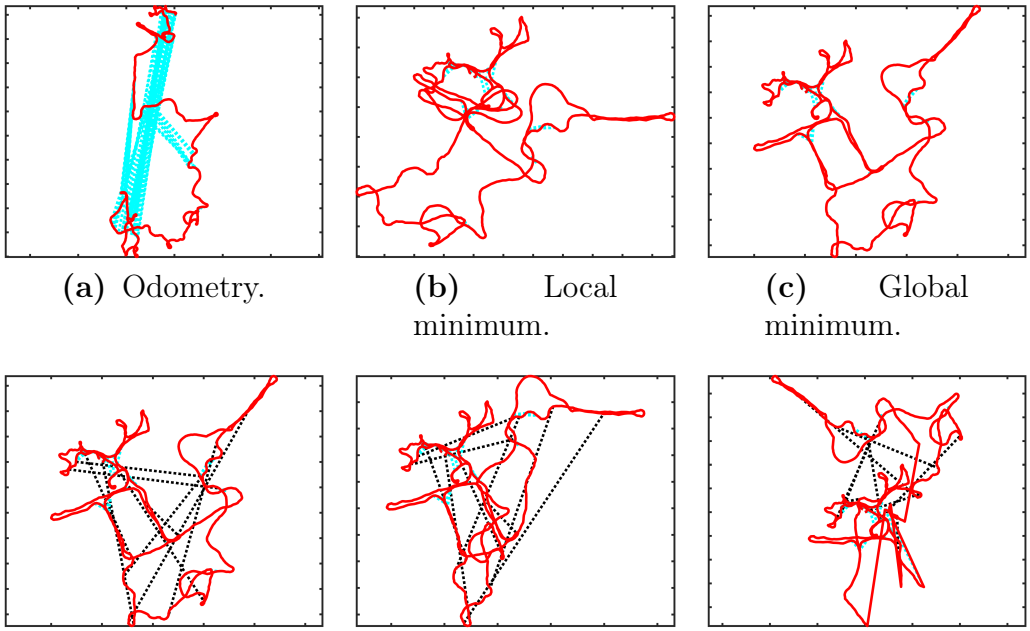
(d) Manhattan (Partial: first 150 independent constraints).

Figure 5.9 An illustration on the accuracy of the COOV metric.

datasets incrementally by iSQP, and record the predicted COOV and the real one in Fig. 5.9. It can be seen from Fig. 5.9 that the predicted COOV almost coincides with the real one, showing the high accuracy of the COOV metric.

Robustness: An Intuitive Example

To get an intuitive understanding on the robustness of iSQP, we use an extremely noisy dataset CSAIL_Pn, with 10 manually added outliers. CSAIL_Pn is a dataset recreated from CSAIL_P [2] with increased noise. The initial odometry of CSAIL_Pn is shown in



(a) Odometry. **(b)** Local minimum. **(c)** Global minimum.

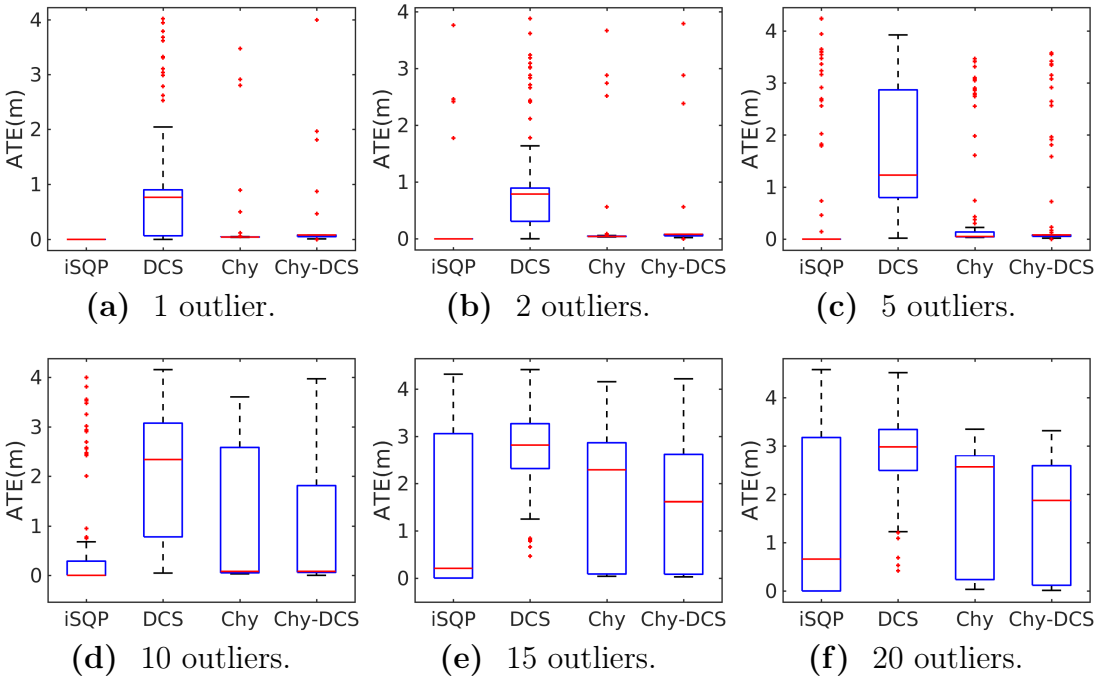
(d) iSQP (10 outliers). **(e)** DCS (10 outliers). **(f)** Cauchy (10 outliers).

Figure 5.10 An intuitive example on the robustness of iSQP.

Fig. 5.10a. In the outlier free case, the Gauss-Newton method initialized by the odometry will converge to a local minimum (Fig. 5.10b). The global minimum can be recovered by using the proposed iSQP, or the Gauss-Newton method bootstrapped by DCS or Cauchy [86], as shown in Fig. 5.10c. Then we add 10 outliers to the CSAIL_Pn dataset to create an even more challenging scenario. In presence of 10 outliers, we find that both DCS (Fig. 5.10e) and Cauchy (Fig. 5.10f) fail while iSQP is still robust (Fig. 5.10d).

Robustness: Monte Carlo Simulation on MITb

The accuracy of the COOV metric is the worst on the MITb dataset in Fig. 5.9. Therefore we opt to use MITb to run a Monte Carlo simulation with respect to different number of outliers to further examine the robustness of iSQP. In specific, we test the robustness in the case of 1, 2, 5, 10, 15, 20 outliers respectively, and for each case we perform a 100 run Monte Carlo simulation. The resulting ATEs of each method are depicted in Fig. 5.11. For



(a) 1 outlier. **(b)** 2 outliers. **(c)** 5 outliers.

(d) 10 outliers. **(e)** 15 outliers. **(f)** 20 outliers.

Figure 5.11 The ATE in 100 run Monte Carlo simulation on the MITb dataset.

comparison, we also use Cauchy+DCS combination (denoted by "Chy+DCS") where Cauchy (denoted by "Chy") is used to bootstrap DCS. The DCS, Chy, and Chy-DCS methods are initialized from the odometry.

The result in Fig. 5.11 shows that: For a moderate number of outliers, i.e., Fig. 5.11a-5.11d, the proposed iSQP performs substantially more robust than the benchmark methods. While in more adverse scenarios, like Fig. 5.11e and Fig. 5.11f, iSQP can achieve comparable robustness, or slightly better in terms of (the median) statistics.

Chapter Six

Conclusion

As a fundamental technique in many robotic tasks, GO was extensively studied in the past decades. By exploiting the inherent sparsity based on sparse linear algebraic techniques, the state-of-the-art solvers these days are capable to solve large GO instances in just a fraction of seconds. However, due to the success of the conventional vertex-based GO algorithms, the fact that most practical GO instances have a low dimension in the cycle space has been largely ignored. As our first major contribution, this thesis looks into the cycle structure of GO, in particular the characterization of the graph structure by a MCB. Although we use PGO as an example, the technique developed in this thesis can be readily extended to other GO instances. In terms of the metrics used in GO, the change of optimal values (COOV) proves to be quite useful, for example in outlier diagnostics, but can only be obtained after solving the problem. Therefore, the very application of COOV always bears a high computational cost, thus the COOV metric has little practical use, particularly in online applications. Besides, the COOV was conceived to be a posterior metric, thus could not be used to guide online decisions. The second contribution in this thesis bridges the gap by developing the exact and approximate equations to forecast COOV in various GO instances. With the novel insights in this thesis, the COOV can be calculated as a prior metric, with computational complexity dropping by magnitudes.

6.1 Contributions

To summarize, this thesis contains the following contributions:

1. We propose CB-PGO, a robust and efficient PGO technique that works in the cycle space of the graph. We characterize the graph sparsity by a MCB, which reduces the numerical complexity and enhances the convergence to the global minimum. We design a tailored MCB algorithm for sparse, positive-integer weighted graphs, which can be used for other cycle based applications as well. We give theoretical insights like LexDijkstra, and working heuristics like pruning vertices of degree two. We provide principled analyses in terms of observability, convergence with respect to cycle bases, and convergence rate for the cycle-based PGO formulation. The claims on the convergence and computational complexity are validated with experiments. We provide an open-source C++ implementation that is freely available to the community to benefit future research in this direction.

2. We develop the equations to forecast the change of optimal values (COOV) in incremental GO settings, for both least squares optimization (i.e., vertex based GO), and minimum norm optimization (i.e., cycle based GO). We prove the exact equation in the linear case, showing that the change of optimal values is solely decided by the solution and the corresponding covariance in the previously solved problem. We extend the result to nonlinear cases via linearization, featuring optimization instances on manifold. Therefore the theory to use change of optimal values as a prior (or pre-calculated) metric has been formally established for incrementally constructed GO instances.

3. We develop three applications to show how the change of optimal values can dramatically change the methodology in GO studies. The first is to detect outliers in the online setting, by making instant decisions based on the COOV metric. We show experimentally that the proposed method is more robust and much faster than M-

estimators. Secondly, we show how the equation to calculate COOV can be used to forecast the optimal cost directly, using an example of aligning two trajectories. The accuracy and computational efficiency are validated with numerical experiments. Last, we show how the COOV metric can be used to design an incremental PGO framework robust against both poor initial values and outliers. The robustness of the proposed framework is demonstrated with experiments on extremely challenging scenarios.

6.2 Discussion and Future Research

6.2.1 MCB Algorithms

In terms of other MCB algorithms, we refer interested readers to the reviewer paper [100]. In theory, the parallelized MCB algorithm described in Chapter 3 has lower complexity over the existing methods in [100]. In practice, our MCB implementation is comparable or even slightly faster than the state-of-the-art implemented by Dimitrios Michail et al. [119]. At this stage, each shortest-path-tree is stored as a chain of backward pointers to the root, which can be compressed as a vector. Therefore we use a dense matrix to describe all-pairs-shortest-path trees (APSP), which requires $O(n^2)$ memory. However, since APSP are chosen to be consistent, it is possible to develop a sparse representation for APSP.

6.2.2 Frobenius Norm based PGO

In (3.1), we use the Logarithm mapping to formulate the cost function. However, other options are possible [83], for example the Frobenius norm used in [42, 140]. For the vertex based PGO, there exists a tight convex relaxation [42] and an efficient Reimannian solver [140] based on the Frobenius norm. For the cycle based PGO (3.3), it is straightforward to derive analogously a Frobenius norm based formulation. A naive solver to this formulation can be designed by SQP similar techniques. However, it can be foreseen that using more

sophisticated designs, for example exploiting convex relaxations and tailored Riemannian solvers, the cycle-based PGO can achieve even better robustness.

6.2.3 Other Cycle based GO Instances

PGO in practice is rather sparse. As a result, this thesis uses PGO as an instance to describe the cycle based approach. However, the characterization of the graph structure by a cycle basis extends to other GO instances, in particular, the importance of using a MCB to improve the numerical efficiency as well as the convergence to the global minimum. Besides, the MCB algorithm developed in this thesis can be readily used to other GO instances, which hopefully will boost the research in other cycle based GO formulations.

6.2.4 Bounds on the COOV Metric

The COOV metric developed for the linear case is exact. However, for nonlinear cases, the COOV metric is derived via linearization, thus is an approximation to the actual COOV. While we show the accuracy of the COOV metric numerically for several typical applications, a theoretical analysis is still required. It is well-known that in the information theory, the Fisher information matrix is bounded by the Cramer Rao bounds, which quantifies the maximum accuracy that an estimate can achieve. In line with Cramer Rao bounds, it is of great interests to find the lower-bound and upper-bound for the approximation errors when using the proposed COOV metric in nonlinear cases.

6.2.5 More Applications based on the COOV Metric

As discussed in Section 4.5, the COOV metric essentially bares the some computational complexity as the information gain, while interprets to different physical implications. Therefore it is clear that every single application based on the information gain can be improved by considering the COOV metric as well. We do not provide any trivial result in this thesis,

but it is an interesting direction to explore. Moreover, although we show several applications in Chapter 5 based on the COOV metric, our exposition is still limited in face of the tremendous possibilities regarding to this novel COOV metric. Therefore we would like to leave this part as future work and open to the community.

Appendix A

Linearization of Cycle Based PGO

In this section, we provide the computational details in terms of how to linearize the CB-PGO formulation in (3.3). Most of the calculations are based on the properties of Lie group. To understand the calculations, readers are expected to know basic operations on Lie group, in particular, the adjoint operation used to switch the position of two matrix exponentials, and the BCH approximation used to concatenate two matrix exponentials. All these concepts are explained in great detail in Chapter 2.2.2.

A.1 Linearization of Cost Function

Denote the error of each edge $\mathbf{k}$ at its estimate $\hat{\mathbf{T}}_{\mathbf{k}}$ to be $\boldsymbol{\eta}_{\mathbf{k}} = \mathbf{Log}(\tilde{\mathbf{T}}_{\mathbf{k}}^{-1} \cdot \hat{\mathbf{T}}_{\mathbf{k}})$. The linearization of the cost function is trivial using the approximate BCH formula:

$$
\begin{aligned}
& \sum_{\mathbf{k}\in\mathcal{E}} \|\mathbf{Log}(\tilde{\mathbf{T}}_{\mathbf{k}}^{-1} \cdot \mathbf{T}_{\mathbf{k}})\|^2_{\boldsymbol{\Sigma}_{\mathbf{k}}} \\
\leftarrow & \sum_{\mathbf{k}\in\mathcal{E}} \|\mathbf{Log}\left(\tilde{\mathbf{T}}_{\mathbf{k}}^{-1} \cdot \hat{\mathbf{T}}_{\mathbf{k}} \cdot \mathbf{Exp}\left(\boldsymbol{\xi}_{\mathbf{k}}\right)\right)\|^2_{\boldsymbol{\Sigma}_{\mathbf{k}}} \\
= & \sum_{\mathbf{k}\in\mathcal{E}} \|\mathbf{Log}\left(\mathbf{Exp}(\boldsymbol{\eta}_{\mathbf{k}}) \cdot \mathbf{Exp}(\boldsymbol{\xi}_{\mathbf{k}})\right)\|^2_{\boldsymbol{\Sigma}_{\mathbf{k}}} \\
\approx & \sum_{\mathbf{k}\in\mathcal{E}} \|\mathbf{Log}\left(\mathbf{Exp}\left(\boldsymbol{\eta}_{\mathbf{k}} + \mathbf{J}_r^{-1}(\boldsymbol{\eta}_{\mathbf{k}})\boldsymbol{\xi}_{\mathbf{k}}\right)\right)\|^2_{\boldsymbol{\Sigma}_{\mathbf{k}}} \\
= & \sum_{\mathbf{k}\in\mathcal{E}} \|\boldsymbol{\eta}_{\mathbf{k}} + \mathbf{J}_r^{-1}(\boldsymbol{\eta}_{\mathbf{k}})\boldsymbol{\xi}_{\mathbf{k}}\|^2_{\boldsymbol{\Sigma}_{\mathbf{k}}} = \|\boldsymbol{\eta} + \mathbf{J}^{-1}\boldsymbol{\xi}\|^2_{\boldsymbol{\Sigma}}
\end{aligned}
$$

where, $\boldsymbol{\eta} = \mathbf{stack}\{\boldsymbol{\eta}_{\mathbf{k}}\}_{\mathbf{k}\in\mathcal{E}}$, $\boldsymbol{\xi} = \mathbf{stack}\{\boldsymbol{\xi}_{\mathbf{k}}\}_{\mathbf{k}\in\mathcal{E}}$, $\mathbf{J} = \mathbf{blkdiag}\{\mathbf{J}_r\left(\boldsymbol{\eta}_{\mathbf{k}}\right)\}_{\mathbf{k}\in\mathcal{E}}$, and $\boldsymbol{\Sigma} = \mathbf{blkdiag}\{\boldsymbol{\Sigma}_{\mathbf{k}}\}_{\mathbf{k}\in\mathcal{E}}$.

A.2 Linearization of Geometric Cycles

Let us take the metric cycle $\boldsymbol{\mathcal{C}}^{\text{lhs}} = \mathbf{I}$ as an example. Recall that $\boldsymbol{\mathcal{C}}^{\text{lhs}} = \mathbf{T}_{\mathbf{1}^c}^{\sigma(\mathbf{1}^c)}\mathbf{T}_{\mathbf{2}^c}^{\sigma(\mathbf{2}^c)}\cdots\mathbf{T}_{\boldsymbol{\lambda}^c}^{\sigma(\boldsymbol{\lambda}^c)}$, where $\mathbf{T}_{\mathbf{1}^c}, \ldots \mathbf{T}_{\boldsymbol{\lambda}^c}$ are the sequential relative poses obtain by a traversal. $\sigma(\mathbf{1}^c), \ldots \sigma(\boldsymbol{\lambda}^c)$ are the relative edge orientations with respect to the traversal, assigned to $+1$ if the traversal uses the edge in the forward direction, and -1 otherwise.

The constraint $\boldsymbol{\mathcal{C}}^{\text{lhs}} = \mathbf{I}$ can be written as $\mathbf{Log}(\boldsymbol{\mathcal{C}}^{\text{lhs}}) = \mathbf{0}$. In the vector space of $\mathfrak{se}(3)$, the first-order Taylor expansion takes the form,

$$\sum_{\mathbf{k}^c=\mathbf{1}^c}^{\boldsymbol{\lambda}^c} \frac{\partial \mathbf{Log}(\boldsymbol{\mathcal{C}}^{\text{lhs}})}{\partial \boldsymbol{\xi}_{\mathbf{k}^c}^T} \cdot \boldsymbol{\xi}_{\mathbf{k}^c} + \mathbf{Log}(\boldsymbol{\mathcal{C}}^{\text{lhs}}) = \mathbf{0}.$$

At a set of given estimates $\hat{\mathbf{T}}_{\mathbf{k}^c}$ ($\mathbf{k}^c = \mathbf{1}^c \cdots \boldsymbol{\lambda}^c$), let's define the error of the geometric cycle to be $\boldsymbol{\beta} = \mathbf{Log}(\boldsymbol{\mathcal{C}}^{\text{lhs}})$. For any $\mathbf{h}^c \in [\mathbf{1}^c, \boldsymbol{\lambda}^c]$, with a bit abuse of notation, we split the geometric cycle into two geometric paths $\boldsymbol{\mathcal{P}}(\mathbf{h}^c)$, $\bar{\boldsymbol{\mathcal{P}}}(\mathbf{h}^c)$:

$$\boldsymbol{\mathcal{P}}(\mathbf{h}^c) = \hat{\mathbf{T}}_{\mathbf{1}^c}^{\sigma(\mathbf{1}^c)} \cdots \hat{\mathbf{T}}_{\mathbf{h}^c}^{\sigma(\mathbf{h}^c)},$$

$$\bar{\boldsymbol{\mathcal{P}}}(\mathbf{h}^c) = \hat{\mathbf{T}}_{(\mathbf{h}+\mathbf{1})^c}^{\sigma((\mathbf{h}+\mathbf{1})^c)} \cdots \hat{\mathbf{T}}_{\boldsymbol{\lambda}^c}^{\sigma(\boldsymbol{\lambda}^c)}.$$

Let $\boldsymbol{\mathcal{P}}(\mathbf{0}^c) = \mathbf{I}$ be a special case. The equality below is trivial

$$\boldsymbol{\mathcal{P}}(\mathbf{h}^c) \cdot \bar{\boldsymbol{\mathcal{P}}}(\mathbf{h}^c) = \mathbf{Exp}(\boldsymbol{\beta}). \tag{A.1}$$

Now let's add a perturbation $\boldsymbol{\xi}_{\mathbf{k}^c}$ to the edge $\mathbf{k}^c$ inside $\boldsymbol{\mathcal{C}}^{\text{lhs}}$ via the exponential mapping,

$$\boldsymbol{\mathcal{C}}^{\text{lhs}} \leftarrow \hat{\mathbf{T}}_{\mathbf{1}^c}^{\sigma(\mathbf{1}^c)} \cdots \left(\hat{\mathbf{T}}_{\mathbf{k}^c} \cdot \mathbf{Exp}(\boldsymbol{\xi}_{\mathbf{k}^c})\right)^{\sigma(\mathbf{k}^c)} \cdots \hat{\mathbf{T}}_{\boldsymbol{\lambda}^c}^{\sigma(\boldsymbol{\lambda}^c)}.$$

Let the perturbed $\boldsymbol{\mathcal{C}}^{\text{lhs}}$ at $\mathbf{k}^c$ be $\boldsymbol{\mathcal{C}}^{\text{lhs}}|_{\mathbf{k}^c}$, which can be compactly written as,

$$\boldsymbol{\mathcal{C}}^{\text{lhs}}|_{\mathbf{k}^c} = \boldsymbol{\mathcal{P}}(\alpha(\sigma(\mathbf{k}^c)))\mathbf{Exp}\left(\sigma(\mathbf{k}^c)\boldsymbol{\xi}_{\mathbf{k}^c}\right)\bar{\boldsymbol{\mathcal{P}}}(\alpha(\sigma(\mathbf{k}^c)))$$

where $\alpha(\sigma(\mathbf{k}^c))$ is a scalar function with respect to $\sigma(\mathbf{k}^c)$,

$$\alpha(\sigma(\mathbf{k}^c)) = \begin{cases} \mathbf{k}^c & \text{if } \sigma(\mathbf{k}^c) = +1 \\ (\mathbf{k}-\mathbf{1})^c & \text{if } \sigma(\mathbf{k}^c) = -1 \end{cases}.$$

Applying (**??**) and considering (A.1), $\boldsymbol{\mathcal{C}}^{\text{lhs}}|_{\mathbf{k}^c}$ can be written as the multiplication of two matrix exponentials, which can be concatenated by the approximate BCH formula, as

$$\begin{aligned}\boldsymbol{\mathcal{C}}^{\text{lhs}}|_{\mathbf{k}^c} &= \mathbf{Exp}\left(\sigma(\mathbf{k}^c)\mathbf{Ad}(\boldsymbol{\mathcal{P}}(\alpha(\sigma(\mathbf{k}^c))))\boldsymbol{\xi}_{\mathbf{k}^c}\right)\mathbf{Exp}\left(\boldsymbol{\beta}\right)\\ &\approx \mathbf{Exp}\left(\sigma(\mathbf{k}^c)\mathbf{J}_l^{-1}(\boldsymbol{\beta})\mathbf{Ad}(\boldsymbol{\mathcal{P}}(\alpha(\sigma(\mathbf{k}^c))))\boldsymbol{\xi}_{\mathbf{k}^c}+\boldsymbol{\beta}\right).\end{aligned}$$

At last, the partial derivative could be calculated as follow,

$$\begin{aligned}\frac{\partial\mathbf{Log}(\boldsymbol{\mathcal{C}}^{\text{lhs}})}{\partial\boldsymbol{\xi}_{\mathbf{k}^c}^T} &= \frac{\partial\mathbf{Log}(\boldsymbol{\mathcal{C}}^{\text{lhs}}|_{\mathbf{k}^c})}{\partial\boldsymbol{\xi}_{\mathbf{k}^c}^T}\\ &=\sigma(\mathbf{k}^c)\mathbf{J}_l^{-1}(\boldsymbol{\beta})\mathbf{Ad}\left(\boldsymbol{\mathcal{P}}(\alpha(\sigma(\mathbf{k}^c)))\right).\end{aligned}$$

It can be seen that $\mathbf{J}_l^{-1}(\boldsymbol{\beta})$ occurs for each derivative $\partial\mathbf{Log}(\boldsymbol{\mathcal{C}}^{\text{lhs}}|_{\mathbf{k}^c})/\partial\boldsymbol{\xi}_{\mathbf{k}^c}$, so the final linearized cycle writes

$$\sum_{\mathbf{k}^c=\mathbf{1}^c}^{\boldsymbol{\lambda}^c}\sigma(\mathbf{k}^c)\mathbf{Ad}\left(\boldsymbol{\mathcal{P}}(\alpha(\sigma(\mathbf{k}^c)))\right)\boldsymbol{\xi}_{\mathbf{k}^c}+\mathbf{J}_l(\boldsymbol{\beta})\boldsymbol{\beta}=\mathbf{0}. \tag{A.2}$$

Without loss of generality, let us assume $\boldsymbol{\mathcal{C}}^{\text{lhs}}=\mathbf{I}$ is the i-th geometric cycle. Then the i-th block row and $\mathbf{k}^c$-th block column of $\mathbf{B}$ writes $\mathbf{B}_{i,\mathbf{k}^c}=\sigma(\mathbf{k}^c)\mathbf{Ad}\left(\boldsymbol{\mathcal{P}}(\alpha(\sigma(\mathbf{k}^c)))\right)$. The i-th block row of $\mathbf{b}$ writes $\mathbf{b}_i=\mathbf{J}_l(\boldsymbol{\beta})\boldsymbol{\beta}$.

For a set of cycles in a cycle basis, we linearize each one of them and assemble the results in the form of $\mathbf{B}\boldsymbol{\xi}+\mathbf{b}=\mathbf{0}$.

Appendix B

Theorems and Proofs on Graph Theory

This section contains proofs of several key results that have been used in the proposed MCB. Theorem 7 is important to understand the restriction of cycle/support vectors to off-tree edges of a spanning tree. Lemma 10 is critical to ensure the correctness of the independence test based on support vectors.

Theorem 7. Let $\mathcal{T}$ be any tree (not necessarily a spanning tree) in $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Define $\bar{\mathcal{C}} \in \mathbb{Z}_2^{\nu}$ to be the incidence vector of $\mathcal{C}$ restricted to the set of off-tree edges in $\mathcal{E}\backslash\mathcal{T}$, i.e.,

$$\mathcal{C} = [\bar{\mathcal{C}}, \tilde{\mathcal{C}}], \text{ where } \bar{\mathcal{C}} \in \mathcal{E}\backslash\mathcal{T}, \ \tilde{\mathcal{C}} \in \mathcal{T}.$$

Then for a collection of cycles $\{\mathcal{C}_i = [\bar{\mathcal{C}}_i, \tilde{\mathcal{C}}_i]\}_{i\in N}$, we have $\mathbf{rank}(\{\mathcal{C}_i\}_{i\in N}) = \mathbf{rank}(\{\bar{\mathcal{C}}_i\}_{i\in N})$.

Proof. Note that $\mathbf{rank}(\{\bar{\mathcal{C}}_i\}_{i\in N}) \leq \mathbf{rank}(\{\mathcal{C}_i\}_{i\in N})$. Then we verify the inequality shall never be reached. Otherwise, there exist at least a sequence of λ_i ($i \in K$), such that $\sum_{i\in K} \lambda_i \bar{\mathcal{C}}_i = \mathbf{0}$ and $\sum_{i\in K} \lambda_i \tilde{\mathcal{C}}_i \neq \mathbf{0}$. As a result, $\mathcal{C}' = \sum_{i\in K} \lambda_i \mathcal{C}_i$ becomes a tree, containing edges only in $\mathcal{T}$, which is impossible. (By composing cycles, the change of the degree for a vertex is even, which implies that each vertex in $\mathcal{C}'$ has an even degree, thus $\mathcal{C}'$ cannot be a tree.) □

Lemma 10. Let $\{\bar{\mathcal{S}}_i\}_{j=k}^{\nu}$ be the support vectors of a collection of independent circuits $\{\mathcal{C}_i\}_{i=1}^{k-1}$. Then a circuit $\mathcal{C}_k$ is linearly independent from $\{\mathcal{C}_i\}_{i=1}^{k-1}$, if and only if there exists a $\bar{\mathcal{S}}_l$ ($k \leq l \leq \nu$) such that $\langle \bar{\mathcal{C}}_k, \bar{\mathcal{S}}_l \rangle = 1$.

Proof. Sufficiency: This is well-known in [50, 99, 100]. Assume $\mathcal{C}_k$ is dependent. Then there is a non-trivial linear combination $\bar{\mathcal{C}}_k = \sum_{i=1}^{k-1} \lambda_i \bar{\mathcal{C}}_i$. As a result, $\langle \bar{\mathcal{C}}_k, \bar{\mathcal{S}}_j \rangle = \sum_{i=1}^{k-1} \lambda_i \langle \bar{\mathcal{C}}_i, \bar{\mathcal{S}}_j \rangle = 0$

holds for all $k \leq j \leq \nu$, which contradicts the existence of $\bar{\mathcal{S}}_l$. Necessity: Assume $\langle \bar{\mathcal{C}}_k, \bar{\mathcal{S}}_j \rangle = 0$ for all $k \leq j \leq \nu$. Then the orthogonality $\langle \bar{\mathcal{C}}_i, \bar{\mathcal{S}}_j \rangle = 0$ would hold for all $1 \leq i \leq k$, $k \leq j \leq \nu$. Since $\mathcal{C}_k$ is independent from $\{\mathcal{C}_i\}_{i=1}^{k-1}$, then $\{\mathcal{C}_i\}_{i=1}^{k}$ are a set of independent circuits. By theorem 7, the restricted circuits $\{\bar{\mathcal{C}}_i\}_{i=1}^{k}$ are also independent. Then $\{\bar{\mathcal{C}}_i\}_{i=1}^{k} \cup \{\bar{\mathcal{S}}_j\}_{j=k}^{\nu}$ are $\nu + 1$ linearly independent vectors for a space of dimension ν, which cannot be true. $\square$

REFERENCES

[1] URL http://www.ipb.uni-bonn.de/datasets/.

[2] URL www.lucacarlone.com/index.php/resources/datasets.

[3] URL https://sourceforge.net/projects/slam-plus-plus/.

[4] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[5] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust map optimization using dynamic covariance scaling. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 62–69. IEEE, 2013.

[6] Pratik Agarwal, Giorgio Grisetti, Gian Diego Tipaldi, Luciano Spinello, Wolfram Burgard, and Cyrill Stachniss. Experimental analysis of dynamic covariance scaling for robust map optimization under bad initial estimates. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3626–3631. IEEE, 2014.

[7] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[8] Edoardo Amaldi, Claudio Iuliano, Tomasz Jurkiewicz, Kurt Mehlhorn, and Romeo Rizzi. Breaking the $\tilde{O}(m^2n)$ barrier for minimum cycle bases. In *European Symposium on Algorithms*, pages 301–312. Springer, 2009.

[9] Edoardo Amaldi, Leo Liberti, Francesco Maffioli, and Nelson Maculan. Edge-swapping algorithms for the minimum fundamental cycle basis problem. *Mathematical Methods of Operations Research*, 69(2):205, 2009.

[10] Edoardo Amaldi, Claudio Iuliano, and Romeo Rizzi. Efficient deterministic algorithms for finding a minimum cycle basis in undirected graphs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 397–410. Springer, 2010.

[11] Patrick R Amestoy, Timothy A Davis, and Iain S Duff. Algorithm 8xx: AMD, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw*, 2003.

[12] Juan Andrade-Cetto and Alberto Sanfeliu. The effects of partial observability when building fully correlated maps. *IEEE Transactions on Robotics*, 21(4):771–777, 2005.

[13] Federica Arrigoni, Beatrice Rossi, and Andrea Fusiello. Spectral synchronization of multiple views in se (3). *SIAM Journal on Imaging Sciences*, 9(4):1963–1990, 2016.

[14] Fang Bai, Shoudong Huang, Teresa Vidal-Calleja, and Qingling Zhang. Incremental SQP method for constrained optimization formulation in SLAM. In *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on*, pages 1–6. IEEE, 2016.

[15] Fang Bai, Teresa Vidal-Calleja, and Shoudong Huang. Robust incremental SLAM under constrained optimization formulation. *IEEE Robotics and Automation Letters*, 3(2):1207–1214, 2018.

[16] Fang Bai, Teresa Vidal-Calleja, Shoudong Huang, and Rong Xiong. Predicting objective function change in pose-graph optimization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 145–152. IEEE, 2018.

[17] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[18] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[19] Timothy D Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[20] Timothy D Barfoot and Paul T Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693, 2014.

[21] Axel Barrau and Silvere Bonnabel. An EKF-SLAM algorithm with consistency properties. *arXiv preprint arXiv:1510.06263*, 2015.

[22] Axel Barrau and Silvere Bonnabel. Invariant kalman filtering. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:237–257, 2018.

[23] Axel Barrau and Silvere Bonnabel. Linear observation systems on groups (I). working paper or preprint, February 2018. URL https://hal-mines-paristech.archives-ouvertes.fr/hal-01671724.

[24] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[25] Ake Bjorck. *Numerical methods for least squares problems*. Siam, 1996.

[26] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[27] Paul T Boggs, Jon W Tolle, and Pyng Wang. On the local convergence of quasi-newton methods for constrained optimization. *SIAM journal on control and optimization*, 20 (2):161–171, 1982.

[28] Guillaume Bourmaud, Rémi Mégret, Audrey Giremus, and Yannick Berthoumieu. Discrete extended kalman filter on lie groups. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5. IEEE, 2013.

[29] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[30] Jesus Briales and Javier Gonzalez-Jimenez. Cartan-Sync: Fast and global SE(d)-synchronization. *IEEE Robotics and Automation Letters*, 2(4):2127–2134, 2017.

[31] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[32] Luca Carlone. A convergence analysis for pose graph optimization via gauss-newton methods. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 965–972. IEEE, 2013.

[33] Luca Carlone and Giuseppe C Calafiore. Convex relaxations for pose graph optimization with outliers. *IEEE Robotics and Automation Letters*, 3(2):1160–1167, 2018.

[34] Luca Carlone and Andrea Censi. From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization. *IEEE Transactions on Robotics*, 30(2):475–492, 2014.

[35] Luca Carlone and Frank Dellaert. Duality-based verification techniques for 2d slam.

In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4589–4596. IEEE, 2015.

[36] Luca Carlone, Rosario Aragues, José A Castellanos, and Basilio Bona. A first-order solution to simultaneous localization and mapping with graphical models. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1764–1771. IEEE, 2011.

[37] Luca Carlone, Rosario Aragues, José A Castellanos, and Basilio Bona. A linear approximation for graph-based simultaneous localization and mapping. *Robotics: Science and Systems VII*, pages 41–48, 2012.

[38] Luca Carlone, Rosario Aragues, José A Castellanos, and Basilio Bona. A fast and accurate approximation for planar pose graph optimization. *The International Journal of Robotics Research*, 33(7):965–987, 2014.

[39] Luca Carlone, Andrea Censi, and Frank Dellaert. Selecting good measurements via ℓ_1 relaxation: A convex approach for robust estimation over graphs. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2667–2674. IEEE, 2014.

[40] Luca Carlone, David M Rosen, Giuseppe Calafiore, John J Leonard, and Frank Dellaert. Lagrangian duality in 3d slam: Verification techniques and optimal solutions. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 125–132. IEEE, 2015.

[41] Luca Carlone, Roberto Tron, Kostas Daniilidis, and Frank Dellaert. Initialization techniques for 3d slam: a survey on rotation estimation and its use in pose graph optimization. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4597–4604. IEEE, 2015.

[42] Luca Carlone, Giuseppe C Calafiore, Carlo Tommolillo, and Frank Dellaert. Planar pose graph optimization: Duality, optimal solutions, and verification. *IEEE Transactions on Robotics*, 32(3):545–565, 2016.

[43] Andrea Cassioli, A Chiavaioli, Costanzo Manes, and Marco Sciandrone. An incremental least squares algorithm for large scale linear classification. *European journal of operational research*, 224(3):560–565, 2013.

[44] Paul Chauchat, Axel Barrau, and Silvere Bonnabel. Invariant smoothing on Lie groups. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1703–1710. IEEE, 2018.

[45] Gregory S Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*, volume 2. Springer Science & Business Media, 2011.

[46] Thomas F Coleman. On characterizations of superlinear convergence for constrained optimization. *Computational Solution of Nonlinear Systems of Equations*, 26:113, 1990.

[47] Mark Cummins and Paul Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[48] Timothy A Davis. *Direct methods for sparse linear systems*, volume 2. Siam, 2006.

[49] Andrew J Davison. Active search for real-time vision. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 66–73. IEEE, 2005.

[50] J.C. de Pina. *Applications of shortest path methods*. Ph.D. Thesis, Universiteit van Amsterdam, 1995.

[51] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[52] Frank Dellaert and Michael Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[53] Frank Dellaert, Justin Carlson, Viorela Ila, Kai Ni, and Charles E Thorpe. Subgraph-preconditioned conjugate gradients for large scale slam. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2566–2571. IEEE, 2010.

[54] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[55] Narsingh Deo, Gurpur Prabhu, and Mukkai S Krishnamoorthy. Algorithms for generating fundamental cycles in a graph. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):26–42, 1982.

[56] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[57] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[58] ET Dixon and Seymour E Goodman. An algorithm for the longest cycle problem. *Networks*, 6(2):139–149, 1976.

[59] Gijs Dubbelman and Brett Browning. COP-SLAM: closed-form online pose-chain optimization for visual SLAM. *IEEE Transactions on Robotics*, 31(5):1194–1213, 2015.

[60] Gijs Dubbelman, Isaac Esteban, and Klamer Schutte. Efficient trajectory bending with applications to loop closure. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4836–4842. IEEE, 2010.

[61] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[62] Debarshi Dutta, Meher Chaitanya, Kishore Kothapalli, and Debajyoti Bera. Applications of ear decomposition to efficient heterogeneous algorithms for shortest path/cycle problems. In *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*, pages 864–873. IEEE, 2017.

[63] Sandro Esquivel, Felix Woelk, and Reinhard Koch. Calibration of a multi-camera rig from non-overlapping views. In *Joint Pattern Recognition Symposium*, pages 82–91. Springer, 2007.

[64] Carlos Estrada, José Neira, and Juan D Tardós. Hierarchical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[65] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[66] Rodrigo Fontecilla, Trond Steihaug, and Richard A Tapia. A convergence theory for a class of quasi-newton methods for constrained optimization. *SIAM Journal on Numerical Analysis*, 24(5):1133–1151, 1987.

[67] John Fox. *Regression diagnostics: An introduction*, volume 79. SAGE Publications, Incorporated, 2019.

[68] Udo Frese, Per Larsson, and Tom Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207, 2005.

[69] Giulia Galbiati, Romeo Rizzi, and Edoardo Amaldi. On the approximability of the minimum strictly fundamental cycle basis problem. *Discrete Applied Mathematics*, 159 (4):187–200, 2011.

[70] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[71] Philip E Gill, Gene H Golub, Walter Murray, and Michael A Saunders. Methods for modifying matrix factorizations. *Mathematics of computation*, 28(126):505–535, 1974.

[72] Alexander Golynski and Joseph D Horton. A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In *Scandinavian Workshop on Algorithm Theory*, pages 200–209. Springer, 2002.

[73] John D Gorman and Alfred O Hero. Lower bounds on parametric estimators with constraints. In *Fourth Annual ASSP Workshop on Spectrum Estimation and Modeling*, pages 223–228. IEEE, 1988.

[74] Matthew C Graham and Jonathan P How. Robust simultaneous localization and mapping via information matrix estimation. In *Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION*, pages 937–944. IEEE, 2014.

[75] Matthew C Graham, Jonathan P How, and Donald E Gustafson. Robust incremental slam with consistency-checking. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 117–124. IEEE, 2015.

[76] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009.

[77] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[78] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, Udo Frese, and Christoph Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 273–278. IEEE, 2010.

[79] Giorgio Grisetti, Rainer Kümmerle, and Kai Ni. Robust optimization of factor graphs by using condensed measurements. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 581–588. IEEE, 2012.

[80] Chao X Guo, Kourosh Sartipi, Ryan C DuToit, Georgios A Georgiou, Ruipeng Li, John O'Leary, Esha D Nerurkar, Joel A Hesch, and Stergios I Roumeliotis. Large-scale cooperative 3d visual-inertial mapping in a manhattan world. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1071–1078. IEEE, 2016.

[81] Richard H Hammack and Paul C Kainen. Graph bases and diagram commutativity. *Graphs and Combinatorics*, 34(4):523–534, 2018.

[82] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[83] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 103(3):267–305, 2013.

[84] David Hartvigsen and Russell Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *SIAM Journal on Discrete Mathematics*, 7(3): 403–418, 1994.

[85] Joseph Douglas Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, 1987.

[86] Gibson Hu, Kasra Khosoussi, and Shoudong Huang. Towards a reliable SLAM back-end. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 37–43. IEEE, 2013.

[87] Shoudong Huang, Yingwu Lai, Udo Frese, and Gamini Dissanayake. How far is SLAM from a linear least squares problem? In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3011–3016. IEEE, 2010.

[88] Peter J Huber. *Robust statistics*. Springer, 2011.

[89] Peter J Huber et al. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[90] Viorela Ila, Josep M Porta, and Juan Andrade-Cetto. Information-based compact pose slam. *IEEE Transactions on Robotics*, 26(1):78–93, 2010.

[91] Viorela Ila, Lukas Polok, Marek Solony, Pavel Smrz, and Pavel Zemcik. Fast covariance recovery in incremental nonlinear least square solvers. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4636–4643. IEEE, 2015.

[92] Viorela Ila, Lukas Polok, Marek Solony, and Pavel Svoboda. SLAM++ 1-A highly efficient and temporally scalable incremental SLAM framework. *The International Journal of Robotics Research*, 36(2):210–230, 2017.

[93] Claude Jauffret. Observability and Fisher information matrix in nonlinear regression. *IEEE Transactions on Aerospace and Electronic Systems*, 43(2):756–759, 2007.

[94] Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and autonomous systems*, 57(12):1198–1210, 2009.

[95] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[96] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[97] Paul C Kainen. On robust cycle bases. *Electronic Notes in Discrete Mathematics*, 11 (430-437):94–95, 2002.

[98] Paul C Kainen. Cycle construction and geodesic cycles with application to the hypercube. *ARS MATHEMATICA CONTEMPORANEA*, 9(1):27–43, 2014.

[99] Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna E Paluch. An $\tilde{O}(m^2n)$ algorithm for minimum cycle basis of graphs. *Algorithmica*, 52(3):333–349, 2008.

[100] Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4): 199–243, 2009.

[101] Kasra Khosoussi, Shoudong Huang, and Gamini Dissanayake. A sparse separable slam back-end. *IEEE Transactions on Robotics*, 32(6):1536–1549, 2016.

[102] Kasra Khosoussi, Matthew Giamou, Gaurav S Sukhatme, Shoudong Huang, Gamini Dissanayake, and Jonathan P How. Reliable graph topologies for SLAM. *International Journal of Robotics Research*, 2018.

[103] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[104] Kurt Konolige. Large-scale map-making. In *AAAI*, pages 457–463, 2004.

[105] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[106] Yasir Latif, César Cadena, and José Neira. Robust loop closing over time for pose graph SLAM. *The International Journal of Robotics Research*, 32(14):1611–1626, 2013.

[107] Charles L Lawson and Richard J Hanson. *Solving least squares problems*, volume 15. Siam, 1995.

[108] Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Robust pose-graph loop-closures with expectation-maximization. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 556–563. IEEE, 2013.

[109] Christian Liebchen and Romeo Rizzi. Classes of cycle bases. *Discrete Applied Mathematics*, 155(3):337–355, 2007.

[110] Minjie Liu, Shoudong Huang, Gamini Dissanayake, and Heng Wang. A convex optimization based approach for pose SLAM problems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1898–1903. IEEE, 2012.

[111] Andrew W Long, Kevin C Wolfe, Michael J Mashner, and Gregory S Chirikjian. The banana distribution is gaussian: A localization study with exponential coordinates. *Robotics: Science and Systems VIII*, 265, 2013.

[112] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

[113] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[114] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[115] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. Methods for non-linear least squares problems. 1999.

[116] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[117] Daniel Martinec and Tomas Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[118] Thomas L Marzetta. A simple derivation of the constrained multiple parameter Cramer-Rao bound. *IEEE Transactions on Signal Processing*, 41(6):2247–2249, 1993.

[119] Kurt Mehlhorn and Dimitrios Michail. Implementing minimum cycle basis algorithms. *Journal of Experimental Algorithmics (JEA)*, 11:2–5, 2007.

[120] Carl D Meyer. *Matrix analysis and applied linear algebra*, volume 71. Siam, 2000.

[121] Michael Montemerlo and Sebastian Thrun. Large-scale robotic 3-d mapping of urban structures. In *Experimental robotics IX*, pages 141–150. Springer, 2006.

[122] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[123] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5): 1147–1163, 2015.

[124] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[125] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *The International Journal of Robotics Research*, 32(7):826–840, 2013.

[126] Edwin Olson, Matthew R Walter, Seth J Teller, and John J Leonard. Single-Cluster Spectral Graph Partitioning for Robotics Applications. In *Robotics: Science and Systems*, pages 265–272, 2005.

[127] Edwin Olson, John Leonard, and Seth Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2262–2269. IEEE, 2006.

[128] Edwin B Olson. *Robust and efficient robotic mapping*. PhD thesis, Massachusetts Institute of Technology, 2008.

[129] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*, volume 30. Siam, 1970.

[130] Jeffrey Russel Peters, Domenica Borra, BE Paden, and Francesco Bullo. Sensor network localization on the group of three-dimensional displacements. *SIAM Journal on Control and Optimization*, 53(6):3534–3561, 2015.

[131] Max Pfingsthorn and Andreas Birk. Generalized graph slam: Solving local and global ambiguities through multimodal and hyperedge constraints. *The International Journal of Robotics Research*, 35(6):601–630, 2016.

[132] Lukas Polok, Viorela Ila, Marek Solony, Pavel Smrz, and Pavel Zemcik. Incremental block cholesky factorization for nonlinear least squares in robotics. In *Robotics: Science and Systems*, pages 328–336, 2013.

[133] Vijaya Ramachandran. *Parallel open ear decomposition with applications to graph biconnectivity and triconnectivity*. Citeseer, 1992.

[134] Santanu Saha Ray. *Graph theory with algorithms and its applications: in applied science and technology*. Springer Science & Business Media, 2012.

[135] William JJ Rey. *Introduction to robust and quasi-robust statistical methods*. Springer Science & Business Media, 2012.

[136] D Rosen and Luca Carlone. Computational enhancements for certifiably correct slam. In *International Conference on Intelligent Robots and Systems (IROS) in the workshop "Introspective Methods for Reliable Autonomy". Google Scholar*, 2017.

[137] David M Rosen, Michael Kaess, and John J Leonard. RISE: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Transactions on Robotics*, 30(5):1091–1108, 2014.

[138] David M Rosen, Charles DuHadway, and John J Leonard. A convex relaxation for approximate global optimization in simultaneous localization and mapping. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5822–5829. IEEE, 2015.

[139] David M Rosen, Luca Carlone, Afonso S. Bandeira, and John J Leonard. A certifiably correct algorithm for synchronization over the special Euclidean group. In *the Algorithmic Foundations of Robotics, The 12th International Workshop on*, 2016.

[140] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. SE-Sync: a certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019.

[141] Wm Joshua Russell, Daniel J Klein, and João P Hespanha. Optimal estimation on the graph cycle space. *IEEE Transactions on Signal Processing*, 59(6):2834–2846, 2011.

[142] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[143] Karin Schermelleh-Engel, Helfried Moosbrugger, and Hans Müller. Evaluating the fit of structural equation models: Tests of significance and descriptive goodness-of-fit measures. *Methods of psychological research online*, 8(2):23–74, 2003.

[144] Dominik Schlegel, Mirco Colosi, and Giorgio Grisetti. Proslam: Graph slam from a programmer's perspective. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[145] Hubert Schwetlick. Gauss-newton-like methods for nonlinear least squares with equality constraints-local convergence and applications. *Statistics: A Journal of Theoretical and Applied Statistics*, 16(2):167–178, 1985.

[146] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435, 2009.

[147] Cyrill Stachniss, John J Leonard, and Sebastian Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 1153–1176. Springer, 2016.

[148] Niko Sünderhauf and Peter Protzel. Towards a robust back-end for pose graph SLAM. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1254–1261. IEEE, 2012.

[149] R A Tapia. Quasi-newton methods for equality constrained optimization: Equivalence of existing methods and a new implementation. In *Nonlinear Programming 3*, pages 125–164. Elsevier, 1978.

[150] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[151] Roberto Tron, Xiaowei Zhou, and Kostas Daniilidis. A survey on rotation optimization in structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 77–85, 2016.

[152] Heng Wang, Gibson Hu, Shoudong Huang, and Gamini Dissanayake. On the structure of nonlinearities in pose graph slam. In *Proc. Robotics: Science Systems*, pages 425–432, 2013.

[153] Jing Wang, Ratan K Ghosh, and Sajal K Das. A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11, 2010.

[154] Yunfeng Wang and Gregory S Chirikjian. Error propagation on the euclidean group with applications to manipulator kinematics. *IEEE Transactions on Robotics*, 22(4): 591–602, 2006.

[155] Zhan Wang and Gamini Dissanayake. Observability analysis of SLAM using fisher information matrix. In *2008 10th International Conference on Control, Automation, Robotics and Vision*, pages 1242–1247. IEEE, 2008.

[156] Hassler Whitney. Non-separable and planar graphs. In *Classic Papers in Combinatorics*, pages 25–48. Springer, 2009.

[157] Teng Zhang, Kanzhi Wu, Jingwei Song, Shoudong Huang, and Gamini Dissanayake. Convergence and consistency analysis for a 3-d Invariant-EKF SLAM. *IEEE Robotics and Automation Letters*, 2(2):733–740, 2017.

[158] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76, 1997.

[159] Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Linear slam: A linear solution to the feature-based and pose graph slam based on submap joining. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 24–30. IEEE, 2013.

[160] Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Linear slam: Linearising the slam problems using submap joining. *Automatica*, 100:231–246, 2019.